mathematics

MDPI

Article

A One-Parameter Memoryless DFP Algorithm for Solving System of Monotone Nonlinear Equations with Application in Image Processing

Najib Ullah [1,2], Abdullah Shah [3], Jamilu Sabi'u [4], Xiangmin Jiao [2], Aliyu Muhammed Awwal [5,6], Nuttapol Pakkaranang [7,*], Said Karim Shah [8] and Bancha Panyanak [9,10]

1 Department of Mathematics, COMSATS University Islamabad, Park Road, Islamabad 45550, Pakistan
2 Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, New York, NY 11794, USA
3 Department of Mathematics, College of Computing and Mathematics, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia
4 Department of Mathematics, Yusuf Maitama Sule University, Kano 700282, Nigeria
5 Department of Mathematics, Faculty of Science, Gombe State University (GSU), Gombe 760214, Nigeria
6 GSU-Mathematics for Innovative Research Group, Gombe State University (GSU), Gombe 760214, Nigeria
7 Mathematics and Computing Science Program, Faculty of Science and Technology, Phetchabun Rajabhat University, Phetchabun 67000, Thailand
8 Department of Physics, Abdul Wali Khan University Mardan, Mardan 23200, Pakistan
9 Research Group in Mathematics and Applied Mathematics, Department of Mathematics, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand
10 Data Science Research Center, Department of Mathematics, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand
* Correspondence: nuttapol.pak@pcru.ac.th

check for updates

Citation: Ullah, N.; Shah, A.; Sabi'u, J.; Jiao, X.; Awwal, A.M.; Pakkaranang, N.; Shah, S.K.; Panyanak, B. A One-Parameter Memoryless DFP Algorithm for Solving System of Monotone Nonlinear Equations with Application in Image Processing. *Mathematics* **2023**, *11*, 1221. https://doi.org/10.3390/math11051221

Academic Editors: Mihai Postolache, Jen-Chih Yao and Yonghong Yao

Received: 31 December 2022
Revised: 21 February 2023
Accepted: 23 February 2023
Published: 2 March 2023

Abstract: In matrix analysis, the scaling technique reduces the chances of an ill-conditioning of the matrix. This article proposes a one-parameter scaling memoryless Davidon–Fletcher–Powell (DFP) algorithm for solving a system of monotone nonlinear equations with convex constraints. The measure function that involves all the eigenvalues of the memoryless DFP matrix is minimized to obtain the scaling parameter's optimal value. The resulting algorithm is matrix and derivative-free with low memory requirements and is globally convergent under some mild conditions. A numerical comparison showed that the algorithm is efficient in terms of the number of iterations, function evaluations, and CPU time. The performance of the algorithm is further illustrated by solving problems arising from image restoration.

Keywords: one-parameter scaling; memoryless DFP algorithm; measure function; convex constraints; image restoration

MSC: 90C30; 90C26; 90C06; 90C56

1. Introduction

The classical quasi–Newton methods are numerically efficient due to their ability to use approximate Jacobian matrices. Consider the following system of monotone nonlinear equations (SMNE) with convex constraints

$$F(x) = 0, \qquad x \in \mathcal{X}, \tag{1}$$

where $x = (x_1, x_2, x_3, ..., x_n)^T$, $\mathcal{X} \subseteq R^n$ is a nonempty closed convex set and $F : R^n \to R^n$ is a continuous and monotone function. The system $F = F_i(i = 1, 2, 3, ..., n)$ is monotone if

$$(F(x) - F(y))^T (x - y) \geq 0, \qquad x, y \in R^n. \tag{2}$$

The solution of system (1) is important in many fields of science and engineering such as control systems [1], signal recovery and image restoration in compressive sensing [2–5], communications and networking [6], data estimation and modeling [7], and geophysics [8].

Newton's method is one of the old techniques for solving a system (1) if the Jacobian matrix is invertible [9]. Initially, Davidon [10] derived a method, which was later on reformulated by Fletcher and Powell [11] known as the Davidon–Fletcher–Powell (DFP) method for approximating the Hessian matrix of the unconstrained optimization problems. The method has been widely used for solving nonlinear optimization problems [12–15].

In recent years, there has been a lot of interest in developing methods for solving the convex-constrained nonlinear monotone problems. Some examples include: Levenberg–Marquardt method [16], scaled trust–region method [17], interior global method [18], Dogleg method [19], Polak–Ribi$\grave{e}$re–Polyak (PRP) method [20], Dai and Yuan method [21], descent derivative–free method [22], projection–based method [23], double projection method [24], multivariate spectral gradient method [25], extension of CG_DESCENT projection method [26], new derivative–free SCG–type projection method [27], partially symmetrical derivative–free Liu–Storey projection method [28], modified spectral gradient projection method [29], efficient generalized conjugate gradient (CG) algorithm [30], modified Fletcher–Reeves method [31], modified decent three–term CG method [32], new hybrid CG projection method [33], self–adaptive three–term CG method [34], efficient three–term CG method [35], derivative–free RMIL CG method [36], and spectral three–term conjugate descent method [37]. A new technique called the inexact Newton method has been introduced by Solodov and Svaiter [9] having an interesting property that the entire system converges to a solution without any assumptions. Later on, Zhou and Toh [38] proved that Solodov and Svaiter method has superlinear convergence under some assumptions. Furthermore, this algorithm was extended by Zhou and Li [39,40] to the Broyden–Fletcher–Goldfarb–Shanno (BFGS) and limited memory BFGS methods. Zhang and Zhou proposed a new method named spectral gradient projection method [41] for the solution of the system (1), which is the combination of spectral gradient method [42] and the projection method [9]. Wang et al. [43] extended the work of Solodov and Svaiter [9] for the solution of monotone equations with convex constraints. Yu et al. [44] extended the spectral gradient projection method for convex constraint problems. Xiao and Zhu [45] extended the CG_DESCENT method [46,47] for large–scale nonlinear convex-constrained monotone equations. Moreover, Awwal et al. [48] derived a new hybrid spectral gradient projection method for the solution of the system (1). Currently, Sabi'u et al. [49] modified the Hager–Zhang CG method by using singular value analysis for solving the system (1).

Inspired by the work of Sabi'u et al. [50,51] for finding some optimal choices of non–negative constants that are involved in some nonlinear CG methods and measure function scaling techniques introduced by Neculai Andrei [52,53]

- We scaled one term of the DFP update formula and found the optimal value of the scaled parameter using the idea of measure function.
- Based on the optimal value of the scaled parameter, we derived a new search direction for the DFP algorithm.
- We proposed a projection-based DFP algorithm for solving large-scale systems of nonlinear monotone equations.
- We provided the global convergence result for the proposed algorithm under some mild assumptions.
- The algorithm is successfully implemented for solving some image restoration problems.

The remainder of this paper is organized as follows. The derivation of the proposed algorithm is given in Section 2. The global convergence of the algorithm is given in Section 3. The numerical results are presented in Section 4. Section 5 contains some applications from the image restoration with its physical explanation. The conclusion is provided in Section 6.

2. One-Parameter Scaled DFP Algorithm

Quasi-Newton schemes are efficient due to their ability to use the Jacobian approximation in the scheme. The DFP update is some of the popular quasi-Newton schemes used for solving large-scale systems of algebraic nonlinear equations. In this section, we present a one-parameter scaled DFP algorithm for solving the system (1). The default iterative formula for the DFP algorithm is as follows

$$x_{k+1} = x_k + \alpha_k d_k, \qquad k = 0, 1, 2, ..., \tag{3}$$

where x_k is the previous iteration, x_{k+1} is the current iteration, α_k is the step–length and d_k is the DFP direction defined as

$$d_k = -H_k F_k, \qquad k = 0, 1, 2, ..., \tag{4}$$

with $F_k = F(x_k)$ and H_k is the DFP matrix at x_k. The updating formula for DFP can be found in [10,54] as

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H y_k} + \frac{s_k s_k^T}{s_k^T y_k}, \qquad k = 0, 1, 2, ..., \tag{5}$$

where $s_k = x_{k+1} - x_k$, and $y_k = F_{k+1} - F_k$. The symmetry of H_{k+1} is directly concern with the symmetry of H_k. One of the well-known properties of the DFP update is that H_{k+1} is positive definite for $s_k^T y_k > 0$ [54]. The main contribution of the scaling technique in the DFP update formula is that it reduces the chances of an ill–conditioning of the matrix H_k for all $k \geq 0$. Now, by multiplying the third term on right-hand side of (5) with a positive scalar γ_k, we have

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H y_k} + \gamma_k \frac{s_k s_k^T}{s_k^T y_k}, \qquad k = 0, 1, 2, ..., \tag{6}$$

where γ_k needs to be determined.

The memoryless concept is applied to the DFP formula to avoid the computation and storage of the matrix H_k at each iteration. In this way, we are replacing the H_k with an identity matrix I_n (where n is dimension of the problem) in (6), so that formula (6) becomes

$$H_{k+1} = I_n - \frac{y_k y_k^T}{y_k^T y_k} + \gamma_k \frac{s_k s_k^T}{s_k^T y_k}, \qquad k = 0, 1, 2, ..., . \tag{7}$$

In addition, we want to utilize the concept of measure function φ on (7), introduced by Byrd and Nocedal [55];

$$\varphi(H_{k+1}) = \mathrm{tr}(H_{k+1}) - \ln(\det(H_{k+1})), \tag{8}$$

where *tr* denotes trace of a positive definite matrix H_{k+1}, *ln* is the natural logarithm, and *det* is the determinant of H_{k+1}. The measure function φ works with *tr* and *det* of H_{k+1} at the same time to modify the quasi–Newton method and to collect information about the behavior of the quasi–Newton method. It is the measure of matrices involving all the eigenvalues of H_{k+1} [56]. If $H_{k+1} = I$, then the function φ is strictly convex [57], while φ becomes unbounded in case of either H_{k+1} is singular or infinite.

Now, the determinant of H_{k+1} can be calculated by the Sherman–Morrison formula [58], i.e.,

$$\det(I + v_1 v_2^T + v_3 v_4^T) = (1 + v_1^T v_2)(1 + v_3^T v_4) - (v_1^T v_4)(v_2^T v_3), \tag{9}$$

to have

$$\det(H_{k+1}) = \gamma_k \frac{y_k^T s_k}{\|y_k\|^2}. \tag{10}$$

Using simple algebra on (7), the trace of H_{k+1} is given by

$$\begin{aligned}\mathrm{tr}(H_{k+1}) &= \mathrm{tr}(I_n) - \mathrm{tr}\left(\frac{y_k y_k^T}{y_k^T y_k}\right) + \gamma_k \mathrm{tr}\left(\frac{s_k s_k^T}{s_k^T y_k}\right)\\ &= \mathrm{tr}(I_n) - \frac{y_k^T y_k}{y_k^T y_k} + \gamma_k \frac{s_k^T s_k}{s_k^T y_k}\\ &= n - \frac{\|y_k\|^2}{\|y_k\|^2} + \gamma_k \frac{\|s_k\|^2}{s_k^T y_k}\\ &= n - 1 + \gamma_k \frac{\|s_k\|^2}{s_k^T y_k}. \end{aligned} \tag{11}$$

Now, putting (10) and (11) into (8), we have

$$\begin{aligned}\varphi(H_{k+1}) &= n - 1 + \gamma_k \frac{\|s_k\|^2}{s_k^T y_k} - \ln\left(\gamma_k \frac{y_k^T s_k}{\|y_k\|^2}\right)\\ &= n - 1 + \gamma_k \frac{\|s_k\|^2}{s_k^T y_k} - \ln(\gamma_k) - \ln(y_k^T s_k) + \ln(\|y_k\|^2). \end{aligned} \tag{12}$$

Differentiating (12) with respect to γ_k, we have

$$\frac{\partial \varphi}{\partial \gamma_k} = \frac{\|s_k\|^2}{s_k^T y_k} - \frac{1}{\gamma_k}. \tag{13}$$

Using the optimality condition on (13), i.e., $\frac{\partial \varphi}{\partial \gamma_k} = 0$, we have

$$\frac{\|s_k\|^2}{s_k^T y_k} = \frac{1}{\gamma_k}, \tag{14}$$

implies that

$$\gamma_k = \frac{s_k^T y_k}{\|s_k\|^2}. \tag{15}$$

Using (15) into (7), we have

$$\begin{aligned} H_{k+1} &= I_n - \frac{y_k y_k^T}{y_k^T y_k} + \frac{s_k^T y_k}{\|s_k\|^2} \frac{s_k s_k^T}{s_k^T y_k}\\ &= I_n - \frac{y_k y_k^T}{y_k^T y_k} + \frac{s_k s_k^T}{\|s_k\|^2}. \end{aligned} \tag{16}$$

Next, using (16) in (4), the search direction is

$$\begin{aligned} d_{k+1} &= -\left(I_n - \frac{y_k y_k^T}{\|y_k\|^2} + \frac{s_k s_k^T}{\|s_k\|^2}\right) F_{k+1}\\ &= -F_{k+1} + \frac{y_k^T F_{k+1}}{\|y_k\|^2} y_k - \frac{s_k^T F_{k+1}}{\|s_k\|^2} s_k. \end{aligned} \tag{17}$$

Furthermore, Solodov and Svaiter [9] proposed a predictor-corrector method, in which

$$z_k = x_k + \alpha_k d_k, \qquad \text{(predictor)} \tag{18}$$

and

$$x_{k+1} = x_k - \lambda_k F(z_k), \qquad \text{(corrector)} \tag{19}$$

where

$$\lambda_k = \frac{F(z_k)^T(x_k - z_k)}{\|F(z_k)\|^2}. \tag{20}$$

In this work, we use an iterative scheme as

$$x_{k+1} = P_\chi[x_k - \xi\lambda_k F(z_k)], \tag{21}$$

where ξ is any positive constant and P is the projection operator on a convex subset χ defined by

$$P_\chi[x] = \arg\min\{\|y - x\| \mid y \in \chi\}, \quad \forall x \in R^n. \tag{22}$$

We recommend that readers read [59] for further details on the advantages and applications of the projection operator. Now, the One-parameter Scaled Memoryless DFP (SMDFP) algorithm for solving system (1) is summarized in Algorithm 1:

Algorithm 1 Scaled Memoryless DFP (SMDFP) Method

Step 0 Given a starting point $x_0 \in R^n$.
Step 1 Choose values for $\xi, \epsilon, \rho > 0$ with $\theta \in (0, 1)$. Compute $d_0 = -F(x_0)$, set $k = 0$.
Step 2 Compute $F(x_k)$, while testing the stopping criteria, i.e. If $\|F_k\| \leq \epsilon$, then stop, otherwise move to the next step.
Step 3 Compute the step size $\alpha_k = \max\{\rho^i; i = 0, 1, 2, \ldots\}$ satisfying the line search

$$-F(x_k + \alpha_k d_k)^T d_k \geq \theta\alpha_k \|F(x_k + \alpha_k d_k)\| \|d_k\|^2. \tag{23}$$

Step 4 Compute the difference $s_k = x_{k+1} - x_k$ and $y_k = F_{k+1} - F_k$.
Step 5 Compute the search direction d_{k+1} using (17).
Step 6 Compute iterative relations x_{k+1} using (21).
Step 7 Set $k = k + 1$ and go to **Step 2**.

Remark 1. *The proposed SMDFP algorithm is a matrix-free and also derivative-free approach. These make the proposed algorithm more efficient in solving large-scale problems.*

3. Convergence Analysis

This section provides the global convergence of the SMDFP algorithm using the following assumptions:

(a) The solution set of system (1) is nonempty and the function F is monotone on R^n, i.e.,

$$(F(x) - F(y))^T(x - y) \geq 0, \quad \forall x, y \in R^n. \tag{24}$$

(b) For a constant $\mu > 0$, the function F is Lipschitz continuous on R^n, i.e.,

$$\|F(x) - F(y)\| \leq \mu\|x - y\|, \quad \forall x, y \in R^n. \tag{25}$$

(c) Let $x^* \in \mathcal{X}$ be the solution of the problem (1) such that $F(x^*) = 0$.

Now, we will proceed with a non–expansive property of projection operator [60].

Lemma 1. *Suppose χ is a nonempty closed and convex subset of R^n. Then the projection operator P can be written as*

$$\|P_\chi[x] - P_\chi[y]\| \leq \|x - y\|, \quad \forall x, y \in R^n, \tag{26}$$

which shows that, P is a Lipschitz continuous on R^n with $\mu = 1$.

Lemma 2. *Our search direction d_{k+1} defined by (17) satisfies the descent condition, i.e.,*

$$F_{k+1}^{\mathrm{T}} d_{k+1} \leq 0, \qquad \forall k \geq 0. \tag{27}$$

Proof. Multiplying (17) by F_{k+1}^{T}, we have

$$\begin{aligned} F_{k+1}^{\mathrm{T}} d_{\mathrm{k+1}} &= F_{k+1}^{\mathrm{T}} \left(-F_{k+1} + \frac{y_k^T F_{k+1}}{\|y_k\|^2} y_k - \frac{s_k^T F_{k+1}}{\|s_k\|^2} s_k \right) \\ &= -\|F_{k+1}\|^2 + \frac{(y_k^T F_{k+1}) F_{k+1}^{\mathrm{T}}}{\|y_k\|^2} y_k - \frac{(s_k^T F_{k+1}) F_{k+1}^{\mathrm{T}}}{\|s_k\|^2} s_k \\ &= -\|F_{k+1}\|^2 + \frac{(y_k^T F_{k+1})^2}{\|y_k\|^2} - \frac{(s_k^T F_{k+1})^2}{\|s_k\|^2} \\ &\leq -\|F_{k+1}\|^2 + \frac{(y_k^T F_{k+1})^2}{\|y_k\|^2} \\ &\leq -\|F_{k+1}\|^2 + \frac{\|y_k\|^2 \|F_{k+1}\|^2}{\|y_k\|^2} \\ &= -\|F_{k+1}\|^2 + \|F_{k+1}\|^2 = 0, \end{aligned} \tag{28}$$

where the second inequality follows from the Cauchy–Schwarz inequality. Hence

$$F_{k+1}^{\mathrm{T}} d_{k+1} \leq 0, \qquad \forall k \geq 0. \tag{29}$$

□

Lemma 3. *Let the assumptions (a), (b), and (c) hold, then the sequences $\{z_k\}$ and $\{x_k\}$ generated by the SMDFP algorithm are bounded. Moreover, we have*

$$\lim_{k \to \infty} \|x_k - z_k\| = 0, \tag{30}$$

and

$$\lim_{k \to \infty} \|x_{k+1} - x_k\| = 0. \tag{31}$$

Proof. Firstly, we will show that the sequences $\{z_k\}$ and $\{x_k\}$ are bounded. Let $x^* \in \mathcal{X}$ be any solution of the problem (1). By monotonicity of F, we get

$$\begin{aligned} \langle F(z_k), x_k - x^* \rangle &= \langle F(z_k), x_k - z_k \rangle + \langle F(z_k), z_k - x^* \rangle \\ &\geq \langle F(z_k), x_k - z_k \rangle + \langle F(x^*), z_k - x^* \rangle \\ &= \langle F(z_k), x_k - z_k \rangle, \end{aligned} \tag{32}$$

and from the definition of z_k and line search (23), we have

$$\begin{aligned} \langle F(z_k), x_k - z_k \rangle &= -\alpha_k \langle F(z_k), d_k \rangle \\ &\geq \theta \alpha_k^2 \|F(z_k)\| \|d_k\|^2 \\ &= \theta \|F(z_k)\| \|x_k - z_k\|^2 > 0. \end{aligned} \tag{33}$$

Now, by using (21) and (26), we get

$$\begin{aligned}\|x_{k+1}-x^*\|^2 &= \|P_X[x_k-\xi\lambda_k F(z_k)]-x^*\|^2\\ &\leq \|x_k-\xi\lambda_k F(z_k)-x^*\|^2\\ &= \|x_k-x^*\|^2-2\xi\lambda_k\langle F(z_k),x_k-x^*\rangle+\|\xi\lambda_k F(z_k)\|^2. \end{aligned}\tag{34}$$

Putting (32) into (34), and then using the value of λ_k from (20), we have

$$\begin{aligned}\|x_{k+1}-x^*\|^2 &\leq \|x_k-x^*\|^2-2\xi\lambda_k\langle F(z_k),x_k-z_k\rangle+\xi^2\lambda_k^2\|F(z_k)\|^2\\ &= \|x_k-x^*\|^2-2\xi\left(\frac{F(z_k)^T(x_k-z_k)}{\|F(z_k)\|^2}\right)\langle F(z_k),x_k-z_k\rangle\\ &+\xi^2\left(\frac{F(z_k)^T(x_k-z_k)}{\|F(z_k)\|^2}\right)^2\|F(z_k)\|^2\\ &= \|x_k-x^*\|^2-2\xi\left(\frac{\langle F(z_k),x_k-z_k\rangle}{\|F(z_k)\|}\right)^2+\xi^2\left(\frac{\langle F(z_k),x_k-z_k\rangle}{\|F(z_k)\|}\right)^2\\ &= \|x_k-x^*\|^2-\xi(2-\xi)\left(\frac{\langle F(z_k),x_k-z_k\rangle}{\|F(z_k)\|}\right)^2. \end{aligned}\tag{35}$$

Now, by using (33) on (35), we have

$$\begin{aligned}\|x_{k+1}-x^*\|^2 &\leq \|x_k-x^*\|^2-\xi(2-\xi)\left(\frac{\theta\|F(z_k)\|\|x_k-z_k\|^2}{\|F(z_k)\|}\right)^2\\ &= \|x_k-x^*\|^2-\xi(2-\xi)\theta^2\|x_k-z_k\|^4. \end{aligned}\tag{36}$$

Hence the sequence $\{\|x_k-x^*\|\}$ is decreasing and convergent. Moreover, the sequence $\{\|x_k\|\}$ is bounded. Furthermore, from (36), we can write

$$\|x_{k+1}-x^*\|^2 \leq \|x_k-x^*\|^2, \quad \forall k\geq 0, \tag{37}$$

and repeating the same process, we get the result that

$$\|x_k-x^*\|^2 \leq \|x_0-x^*\|^2, \quad \forall k\geq 0. \tag{38}$$

Moreover, from assumption (b), we have

$$\|F(x_k)\| = \|F(x_k)-F(x^*)\| \leq \mu\|x_k-x^*\| \leq \mu\|x_0-x^*\|. \tag{39}$$

We suppose that $\mu\|x_0-x^*\| = \omega$, then the sequence $\{F(x_k)\}$ is bounded, i.e.,

$$\|F(x_k)\| \leq \omega, \quad \forall k\geq 0. \tag{40}$$

By the Cauchy–Schwarz inequality, monotonicity of F and inequality (33), it holds that

$$0<\theta\|F(z_k)\|\|x_k-z_k\|^2 \leq \langle F(z_k),x_k-z_k\rangle \leq \|F(z_k)\|\|x_k-z_k\|. \tag{41}$$

From the inequality (41), it implies that

$$\theta\|x_k-z_k\| \leq 1, \tag{42}$$

which shows that the sequence $\{z_k\}$ is also bounded. Moreover, from inequality (36), it follows that

$$\xi(2-\xi)\theta^2 \sum_{k=1}^{\infty} \|x_k - z_k\|^4 \leq \sum_{k=1}^{\infty} \left(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2 \right) \leq \|x_0 - x_*\| < \infty, \tag{43}$$

which implies that

$$\lim_{k\to\infty} \|x_k - z_k\| = 0. \tag{44}$$

Since $x_k \in X$, so by using (26) and then putting the value of λ_k from (20), we have

$$\begin{aligned} \|x_{k+1} - x_k\| &= \|P_X[x_k - \xi\lambda_k F(z_k)] - x^*\| \\ &\leq \|x_k - \xi\lambda_k F(z_k) - x_k\| \\ &= \|\xi\lambda_k F(z_k)\| \\ &= \xi \left(\frac{\langle F(z_k), x_k - z_k \rangle}{\|F(z_k)\|^2} \right) \|F(z_k)\| \\ &= \xi \frac{\langle F(z_k), x_k - z_k \rangle}{\|F(z_k)\|}. \end{aligned} \tag{45}$$

Now using the Cauchy–Schwarz inequality on (45), we have

$$\begin{aligned} \|x_{k+1} - x_k\| &\leq \xi \frac{\|F(z_k)\| \|x_k - z_k\|^2}{\|F(z_k)\|} \\ &= \xi \|x_k - z_k\|^2, \quad \forall k \geq 0. \end{aligned} \tag{46}$$

Thus, by using (30), we have

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0.$$

Hence, the proof of (31) is completed. □

Remark 2. *Let the sequence $\{x_k\}$ be generated by the SMDFP method. Then, using (18) on (44), we have*

$$\begin{aligned} \lim_{k\to\infty} \|z_k - x_k\| &= \lim_{k\to\infty} \|x_k + \alpha_k d_k - x_k\| \\ &= \lim_{k\to\infty} \alpha_k \|d_k\| = 0, \quad \forall k \geq 0. \end{aligned} \tag{47}$$

Lemma 4. *The direction generated by the SMDFP algorithm is bounded. That is*

$$\|d_{k+1}\| \leq q, \quad \forall k \geq 0, \tag{48}$$

where q is some positive constant.

Proof. It follows from (17) that

$$\begin{aligned} \|d_{k+1}\| &= \left\| -F_{k+1} + \frac{y_k^T F_{k+1}}{\|y_k\|^2} y_k - \frac{s_k^T F_{k+1}}{\|s_k\|^2} s_k \right\| \\ &\leq \|F_{k+1}\| + \left\| \frac{y_k^T F_{k+1}}{\|y_k\|^2} y_k \right\| + \left\| \frac{s_k^T F_{k+1}}{\|s_k\|^2} s_k \right\| \\ &\leq \|F_{k+1}\| + \|F_{k+1}\| + \|F_{k+1}\| \\ &= 3\|F_{k+1}\| \leq 3\omega = q, \end{aligned} \tag{49}$$

where the second inequity follows from Cauchy–Schwarz inequality. Thus,

$$\|d_{k+1}\| \leq q, \quad \forall k \geq 0. \tag{50}$$

□

Theorem 1. *Let the sequences* $\{z_k\}$ *and* $\{x_k\}$ *be generated by the SMDFP algorithm, then*

$$\liminf_{k\to\infty} \|F_k\| = 0. \tag{51}$$

Proof. To prove that (51) holds, we consider the following two cases;

Case 1. Suppose the sequence $\{x_k\}$ is generated by the SMDFP method. Then we have

$$\lim_{k\to\infty} \alpha_k \|d_k\| = 0. \tag{52}$$

Assume that if

$$\liminf_{k\to\infty} \|d_k\| = 0,$$

then

$$\liminf_{k\to\infty} \|F_k\| = 0.$$

Then by continuity property of F, there will be some accumulation point x^* in sequence $\{x_k\}$ such that $F(x^*) = 0$. Since $\{\|x_k - x^*\|\}$ is going to converge and $\{x_k\}$ has an accumulation point x^*, $\{x_k\}$ is going to converge to x^*.

Case 2. Suppose (51) is not true, and then there exists some positive constant δ such that

$$\|F_k\| \geq \delta > 0, \quad \forall k \geq 0. \tag{53}$$

By using Cauchy–Schwarz inequality on descent condition, we have

$$\|F_k\|\|d_k\| \geq -{F_k}^T d_k \geq \|F_k\|^2 \geq 0, \quad \forall k \geq 0, \tag{54}$$

which implies that

$$\|d_k\| \geq \|F_k\| > 0, \quad \forall k \geq 0. \tag{55}$$

Using (52) and (55), we have

$$\liminf_{k\to\infty} \|d_k\| > 0. \tag{56}$$

Now, by using (52) and (56), we have

$$\lim_{k\to\infty} \alpha_k = 0. \tag{57}$$

Since (57) is true, by the definition of α_k, $\rho^{-1}\alpha_k$ does not satisfy the line search, i.e.,

$$-F(x_k + \alpha_k\rho^{-1}d_k)^T d_k < \theta\alpha_k\rho^{-1}\left\|F(x_k + \alpha_k\rho^{-1}d_k)\right\|\|d_k\|^2, \tag{58}$$

and from the boundedness of $\{x_k\}$ and $\{d_k\}$, we can choose a sub–sequence such that k approaches to infinity in the above inequality results, then (58) becomes

$$-F(x^*)^T \bar{d} < 0. \tag{59}$$

Moreover, k approaches to ∞ in (29), which implies that

$$-F(x^*)^T\bar{d} \geq 0. \tag{60}$$

From (59) and (60), we concluded that it is a clear contradiction to each other. Hence,

$$\liminf_{k\to\infty}\|F_k\| = 0,$$

is true and the proof is complete. □

4. Numerical Experimentation

In this section, we perform some numerical experiments to validate the SMDFP algorithm by comparing the computed results with the conjugate gradient hybrid (CGH) method [61] and with the generalized hybrid CGPM–based (GHCGP) method [62]. All algorithms are written in Matlab R2014 on an HP CORE i5 Intel 8th Gen personal computer. In our experiment with the uniform stopping condition, we used the published initial values for the comparison algorithms. However, we used $\theta = 0.0001$ and $\rho = 0.9$ in the SMDFP algorithm. The numerical simulations are stopped either exceeding the dimension or $\|F(x_k)\| \leq 10^{-11}$. For the following problems see [63–68].

Problem 1 ([63,64]). *Set* $\mathcal{X} = \mathbb{R}^n_+$ *and function* $F(x)$ *is described as*
$F_i(x) = \exp(x_i) - 1, \quad for \quad i = 1, 2, 3, \ldots, n.$

Problem 2 ([65]). *Set* $\mathcal{X} = \mathbb{R}^n_+$ *and function* $F(x)$ *is described as*
$F_1(x) = \cos(x_1) + 3x_1 + 8\exp(x_2) - 9,$
$F_i(x) = \cos(x_i) + 3x_i + 8\exp(x_{i-1}) - 9, \quad for \quad i = 2, 3, 4, \ldots, n.$

Problem 3 ([66]). *Set* $\mathcal{X} = \mathbb{R}^n_+$ *and function* $F(x)$ *is described as*
$F_i(x) = x_i - 0.1x^2_{i+1}, \quad for \quad i = 2, 3, 4, \ldots, n-1,$
$F_n(x) = x_n - 0.1x^2_1.$

Problem 4 ([67]). *Set* $\mathcal{X} = \mathbb{R}^n_+$ *and function* $F(x)$ *is described as*
$F_i(x) = x_i\cos(x_i - \frac{1}{n}) - x_n\left[\sin(x_i) - 1 - (x_i - 1)^2 - \frac{1}{n}\sum_{i=1}^n x_i\right],$
for $\quad i = 1, 2, 3, \ldots, n.$

Problem 5 ([68]). *Set* $\mathcal{X} = \mathbb{R}^n_+$ *and function* $F(x)$ *is described as*
$F_i(x) = x_i^2 + 10x_i, \quad for \quad i = 1, 2, 3, \ldots, n.$

Problem 6 ([63]). *Set* $\mathcal{X} = \mathbb{R}^n_+$ *and function* $F(x)$ *is described as*
$F_i(x) = log(\|x_i\| + 1) - \frac{x_i}{n}, \quad for \quad i = 1, 2, 3, \ldots, n.$

Problem 7 ([63,64]). *Set* $\mathcal{X} = \mathbb{R}^n_+$ *and function* $F(x)$ *is described as*
$F_i(x) = 2x_i - \sin\|x_i\|, \quad for \quad i = 1, 2, 3, \ldots, n.$

Problem 8 ([63]). *Set* $\mathcal{X} = \mathbb{R}^n_+$ *and function* $F(x)$ *is described as*
$F_1(x) = \exp(x_1) - 1,$
$F_i(x) = \exp(x_i) + x_i - 1, \quad for \quad i = 2, 3, 4, \ldots, n.$

In Tables 1–8, we used the initial points $x_1 = (1, 1, \ldots, 1)$, $x_2 = (1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{n})$, $x_3 = (0.1, 0.1, \ldots, 0.1)$, $x_4 = (\frac{1}{n}, \frac{2}{n}, \ldots, 1)$, $x_5 = (1 - \frac{1}{n}, 1 - \frac{2}{n}, \ldots, 0)$, $x_6 = (-1, -1, \ldots, -1)$, $x_7 = (n - \frac{1}{n}, n - \frac{2}{n}, \ldots, n - 1)$ and $x_8 = (\frac{1}{2}, 1, \frac{2}{3}, \ldots, \frac{2}{n})$. Moreover, the terms ITER, FEV, CPUT, and NORM stand for the number of iterations, the number of function evaluations, CPU times (in seconds), and the norm of the function evaluations, respectively. The failure of a certain algorithm is denoted by "_". We further show the numerical performance of

the SMDFP algorithm along with the CGH [61] and GHCGP [62] methods in terms of the number of iterations, function evaluations, CPU times, and error estimation.

In Tables 1 and 2, the SMDFP algorithm has fewer iterations and function evaluations, shorter CPU times, and smaller errors compared to CGH [61] and GHCGP [62] methods. However, the SMDFP algorithm has more number of iterations, function evaluations, shorter CPU times, and smaller errors than the two compared methods in Tables 3, 5 and 6. The CGH method failed for Problems 4 and 7, as shown in Tables 4 and 7. Further from Table 8, it is noted that the GHCGP algorithm failed for Problem 8. It is concluded that the overall performance of the SMDFP algorithm is more efficient than both the CGH approach and GHCGP algorithm in terms of the number of iterations, function evaluations, CPU times, and error estimation, as shown in Tables 1–8.

Table 1. Numerical comparison of the SMDFP, CGH [61], and GHCGP [62] methods.

Problem 1		**SMDFP**				**CGH**				**GHCGP**			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
1000	x_1	1	3	0.035122	0	130	1680	2.236476	8.41×10^{-9}	30	61	0.143056	7.2×10^{-9}
	x_2	1	3	0.014751	0	1	3	0.007694	0	31	64	0.030084	9.34×10^{-9}
	x_3	1	3	0.00864	0	1	3	0.004303	0	32	68	0.028365	9.12×10^{-9}
	x_4	1	3	0.004609	0	1	3	0.003516	0	31	66	0.026267	6.23×10^{-9}
	x_5	1	3	0.004232	0	1	3	0.003499	0	1	3	0.003961	0
	x_6	1	3	0.0045	0	107	1392	0.141467	8.73×10^{-9}	30	61	0.02529	8.66×10^{-9}
	x_7	1	3	0.004224	0	95	1236	0.147844	9.5×10^{-9}	29	59	0.025136	5.33×10^{-9}
	x_8	1	3	0.004219	0	1	3	0.003311	0	1	3	0.004024	0
5000	x_1	1	3	0.01969	0	134	1732	1.826228	8.17×10^{-9}	31	63	0.282029	8.05×10^{-9}
	x_2	1	3	0.01655	0	1	3	0.007681	0	33	68	0.460068	5.22×10^{-9}
	x_3	1	3	0.013203	0	1	3	0.006739	0	34	72	0.461073	5.1×10^{-9}
	x_4	1	3	0.03535	0	1	3	0.006651	0	32	68	0.605644	6.97×10^{-9}
	x_5	1	3	0.011906	0	1	3	0.020484	0	1	3	0.013328	0
	x_6	1	3	0.016778	0	111	1444	0.63665	8.47×10^{-9}	31	63	0.548416	9.68×10^{-9}
	x_7	1	3	0.01269	0	99	1288	0.62048	9.22×10^{-9}	30	61	0.629381	5.95×10^{-9}
	x_8	1	3	0.015683	0	1	3	0.006559	0	1	3	0.017017	0
15,000	x_1	1	3	0.01634	0	135	1745	1.434341	9.38×10^{-9}	32	65	0.986364	5.69×10^{-9}
	x_2	1	3	0.017012	0	1	3	0.007016	0	33	68	1.136386	7.38×10^{-9}
	x_3	1	3	0.021289	0	1	3	0.007927	0	34	72	1.153497	7.21×10^{-9}
	x_4	1	3	0.014781	0	1	3	0.009047	0	32	68	1.039588	9.86×10^{-9}
	x_5	1	3	0.015541	0	1	3	0.006813	0	1	3	0.025195	0
	x_6	1	3	0.015866	0	112	1457	1.40839	9.72×10^{-9}	32	65	1.296489	6.85×10^{-9}
	x_7	1	3	0.020727	0	101	1314	1.66346	8.59×10^{-9}	30	61	1.407296	8.42×10^{-9}
	x_8	1	3	0.014068	0	1	3	0.014423	0	1	3	0.02878	0
50,000	x_1	1	3	0.035269	0	139	1797	8.660152	9.1×10^{-9}	33	67	2.698926	6.36×10^{-9}
	x_2	1	3	0.036429	0	1	3	0.026383	0	34	70	2.620559	8.25×10^{-9}
	x_3	1	3	0.04011	0	1	3	0.02646	0	35	74	2.367671	8.06×10^{-9}
	x_4	1	3	0.033272	0	1	3	0.029999	0	34	72	2.239652	5.51×10^{-9}
	x_5	1	3	0.032413	0	1	3	0.158757	0	1	3	0.080205	0
	x_6	1	3	0.069492	0	116	1509	39.25009	9.44×10^{-9}	33	67	2.284177	7.65×10^{-9}
	x_7	1	3	0.031998	0	105	1366	8.375691	8.34×10^{-9}	31	63	2.439808	9.42×10^{-9}
	x_8	1	3	0.042604	0	1	3	0.024109	0	1	3	0.0487	0
100,000	x_1	1	3	0.072912	0	141	1823	15.78274	8.48×10^{-9}	33	67	2.800037	9×10^{-9}
	x_2	1	3	0.090527	0	1	3	0.044907	0	35	72	3.262603	5.84×10^{-9}
	x_3	1	3	0.078645	0	1	3	0.045901	0	36	76	3.094576	5.7×10^{-9}
	x_4	1	3	0.075346	0	1	3	0.054425	0	34	72	2.737321	7.79×10^{-9}
	x_5	1	3	0.126657	0	1	3	0.051759	0	1	3	0.128415	0
	x_6	1	3	0.071292	0	118	1535	14.85632	8.8×10^{-9}	34	69	3.082105	5.41×10^{-9}
	x_7	1	3	0.091889	0	106	1379	12.69957	9.57×10^{-9}	32	65	2.589503	6.66×10^{-9}
	x_8	1	3	0.111511	0	1	3	0.03606	0	1	3	0.084673	0

Table 2. Numerical comparison of the SMDFP, CGH [61], and GHCGP [62] methods.

Problem 2		**SMDFP**				**CGH**				**GHCGP**			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
1000	x_1	1	3	0.005111	0	1	3	0.201878	0	13	55	0.057429	1.93×10^{-9}
	x_2	1	3	0.004934	0	1	3	0.004667	0	12	50	0.02292	2.44×10^{-9}
	x_3	1	3	0.004862	0	1	3	0.004516	0	14	61	0.02838	2×10^{-9}
	x_4	1	3	0.00481	0	1	3	0.004824	0	15	69	0.030771	1.66×10^{-9}
	x_5	1	3	0.004774	0	1	3	0.004738	0	12	55	0.023997	2.48×10^{-9}
	x_6	1	3	0.004873	0	1	3	0.004792	0	12	50	0.022978	1.78×10^{-9}
	x_7	1	3	0.004563	0	1	3	0.004552	0	11	45	0.01946	1.91×10^{-9}
	x_8	1	3	0.005762	0	1	3	0.004795	0	14	67	0.028311	2.31×10^{-9}
5000	x_1	1	3	0.018152	0	1	3	0.016958	0	13	55	1.230151	4.31×10^{-9}
	x_2	1	3	0.01962	0	1	3	0.054983	0	12	50	0.988219	5.45×10^{-9}
	x_3	1	3	0.021361	0	1	3	0.071534	0	14	61	2.208304	4.46×10^{-9}
	x_4	1	3	0.049504	0	1	3	0.028522	0	15	69	2.334171	3.72×10^{-9}
	x_5	1	3	0.028837	0	1	3	0.026218	0	12	55	0.976465	5.54×10^{-9}
	x_6	1	3	0.014489	0	1	3	0.023154	0	12	50	1.120086	3.97×10^{-9}
	x_7	1	3	0.020818	0	1	3	0.036165	0	11	45	1.011787	4.27×10^{-9}
	x_8	1	3	0.035658	0	1	3	0.023837	0	14	67	1.645371	5.18×10^{-9}
15,000	x_1	1	3	0.024338	0	1	3	0.042924	0	13	55	1.554233	6.09×10^{-9}
	x_2	1	3	0.025928	0	1	3	0.0635	0	12	50	2.311009	7.71×10^{-9}
	x_3	1	3	0.028652	0	1	3	0.033185	0	14	61	1.574	6.31×10^{-9}
	x_4	1	3	0.027035	0	1	3	0.049415	0	15	69	1.408958	5.26×10^{-9}
	x_5	1	3	0.028536	0	1	3	0.231511	0	12	55	1.176572	7.84×10^{-9}
	x_6	1	3	0.020113	0	1	3	0.03914	0	12	50	1.131856	5.62×10^{-9}
	x_7	1	3	0.033993	0	1	3	0.086277	0	11	45	1.110647	6.04×10^{-9}
	x_8	1	3	0.024769	0	1	3	0.102031	0	14	67	1.089978	7.32×10^{-9}
50,000	x_1	1	3	0.100548	0	1	3	0.081128	0	14	59	1.840833	1.63×10^{-9}
	x_2	1	3	0.059629	0	1	3	0.107223	0	13	54	1.495333	2.07×10^{-9}
	x_3	1	3	0.060439	0	1	3	0.089753	0	15	65	2.181382	1.69×10^{-9}
	x_4	1	3	0.066735	0	1	3	0.083639	0	16	73	2.307287	1.41×10^{-9}
	x_5	1	3	0.057437	0	1	3	0.09488	0	13	59	1.942567	2.1×10^{-9}
	x_6	1	3	0.060867	0	1	3	0.111695	0	13	54	1.833159	1.51×10^{-9}
	x_7	1	3	0.064029	0	1	3	0.156286	0	12	49	1.23558	1.62×10^{-9}
	x_8	1	3	0.065988	0	1	3	0.326879	0	15	71	2.245318	1.96×10^{-9}
100,000	x_1	1	3	0.119022	0	1	3	0.172609	0	14	59	2.501767	2.31×10^{-9}
	x_2	1	3	0.127509	0	1	3	0.314452	0	13	54	2.947286	2.93×10^{-9}
	x_3	1	3	0.132327	0	1	3	0.224496	0	15	65	2.886944	2.4×10^{-9}
	x_4	1	3	0.120837	0	1	3	0.28043	0	16	73	3.275997	2×10^{-9}
	x_5	1	3	0.163017	0	1	3	0.218818	0	13	59	2.979261	2.97×10^{-9}
	x_6	1	3	0.147234	0	1	3	0.317258	0	13	54	1.977479	2.13×10^{-9}
	x_7	1	3	0.114379	0	1	3	0.164473	0	12	49	2.349758	2.29×10^{-9}
	x_8	1	3	0.155111	0	1	3	0.164645	0	15	71	2.860774	2.78×10^{-9}

Table 3. Numerical comparison of the SMDFP, CGH [61], and GHCGP [62] methods.

Problem 3		**SMDFP**				**CGH**				**GHCGP**			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
1000	x_1	27	352	0.117364	7.01×10^{-9}	104	1353	0.318394	9.22×10^{-9}	32	65	0.061522	9.06×10^{-9}
	x_2	28	365	0.123548	7.87×10^{-9}	2	16	0.006879	0	34	69	0.031974	5.68×10^{-9}
	x_3	29	378	0.120701	6.79×10^{-9}	2	16	0.00681	0	35	71	0.031623	5.47×10^{-9}
	x_4	30	391	0.134075	9.73×10^{-9}	2	16	0.007006	0	36	73	0.032578	8.43×10^{-9}
	x_5	33	430	0.146445	7.54×10^{-9}	9	73	0.018668	9.97×10^{-9}	39	79	0.036347	8.49×10^{-9}
	x_6	26	339	0.112825	7.14×10^{-9}	102	1327	0.297255	8.12×10^{-9}	31	63	0.028116	8.15×10^{-9}
	x_7	24	313	0.100542	6.87×10^{-9}	94	1223	0.285619	9.71×10^{-9}	29	59	0.0284	6.01×10^{-9}
	x_8	-	-	-	-	-	-	-	-	-	-	-	-
5000	x_1	28	365	2.139238	6.82×10^{-9}	108	1405	13.72612	8.95×10^{-9}	34	69	2.214968	5.06×10^{-9}
	x_2	29	378	2.203859	7.66×10^{-9}	2	16	0.095661	0	35	71	2.019577	6.35×10^{-9}
	x_3	30	391	2.179331	6.61×10^{-9}	2	16	0.100748	0	36	73	1.722336	6.12×10^{-9}
	x_4	31	404	1.693619	9.47×10^{-9}	2	16	0.168732	0	37	75	1.493552	9.43×10^{-9}
	x_5	34	443	1.419028	7.34×10^{-9}	9	73	0.611932	2.23×10^{-9}	40	81	3.0625	9.49×10^{-9}
	x_6	27	352	0.746712	6.95×10^{-9}	105	1366	7.824971	9.71×10^{-9}	32	65	2.277345	9.11×10^{-9}
	x_7	25	326	0.761351	6.69×10^{-9}	98	1275	5.246163	9.43×10^{-9}	30	61	1.597036	6.72×10^{-9}
	x_8	-	-	-	-	-	-	-	-	-	-	-	-

Table 3. *Cont.*

Problem 3		SMDFP				CGH				GHCGP			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
10,000	x_1	28	365	1.48354	9.65×10^{-9}	110	1431	6.355567	8.34×10^{-9}	34	69	1.958965	7.16×10^{-9}
	x_2	30	391	1.390034	4.71×10^{-9}	2	16	0.074192	0	35	71	2.865596	8.98×10^{-9}
	x_3	30	391	1.431256	9.35×10^{-9}	2	16	0.067912	0	36	73	2.382778	8.65×10^{-9}
	x_4	32	417	1.53683	5.83×10^{-9}	2	16	0.052701	0	38	77	2.529481	6.66×10^{-9}
	x_5	35	456	1.352757	4.52×10^{-9}	9	73	0.584748	3.15×10^{-9}	41	83	3.048691	6.71×10^{-9}
	x_6	27	352	0.924018	9.83×10^{-9}	107	1392	5.41707	9.05×10^{-9}	33	67	2.122735	6.44×10^{-9}
	x_7	25	326	0.898503	9.46×10^{-9}	100	1301	4.89272	8.78×10^{-9}	30	61	1.577585	9.5×10^{-9}
	x_8	-	-	-	-	-	-	-	-	-	-	-	-
50,000	x_1	29	378	4.140748	9.39×10^{-9}	113	1470	11.60575	9.98×10^{-9}	35	71	2.654783	8.01×10^{-9}
	x_2	31	404	4.157986	4.58×10^{-9}	2	16	0.120349	0	37	75	2.537268	5.02×10^{-9}
	x_3	31	404	3.655449	9.1×10^{-9}	2	16	0.140539	0	37	75	2.57247	9.67×10^{-9}
	x_4	33	430	3.891895	5.67×10^{-9}	2	16	0.14212	0	39	79	2.641795	7.45×10^{-9}
	x_5	36	469	4.213919	4.39×10^{-9}	9	73	0.730919	7.05×10^{-9}	42	85	3.088176	7.51×10^{-9}
	x_6	28	365	3.438646	9.56×10^{-9}	111	1444	10.96663	8.78×10^{-9}	34	69	2.17434	7.2×10^{-9}
	x_7	26	339	3.296482	9.2×10^{-9}	104	1353	9.567916	8.53×10^{-9}	32	65	1.841247	5.31×10^{-9}
	x_8	-	-	-	-	-	-	-	-	-	-	-	-
100,000	x_1	30	391	7.675792	5.78×10^{-9}	115	1496	18.44836	9.29×10^{-9}	36	73	3.191328	5.66×10^{-9}
	x_2	31	404	7.127316	6.48×10^{-9}	2	16	0.173612	0	37	75	3.53366	7.1×10^{-9}
	x_3	32	417	7.365149	5.6×10^{-9}	2	16	0.260983	0	38	77	3.179414	6.84×10^{-9}
	x_4	33	430	7.323262	8.02×10^{-9}	2	16	0.238622	0	40	81	3.479403	5.27×10^{-9}
	x_5	36	469	7.954414	6.21×10^{-9}	9	73	1.261594	9.97×10^{-9}	43	87	4.137574	5.31×10^{-9}
	x_6	29	378	6.641638	5.89×10^{-9}	113	1470	21.35079	8.19×10^{-9}	35	71	2.87259	5.09×10^{-9}
	x_7	27	352	6.081961	5.66×10^{-9}	105	1366	15.93775	9.79×10^{-9}	32	65	2.929885	7.51×10^{-9}
	x_8	-	-	-	-	-	-	-	-	-	-	-	-

Table 4. Numerical comparison of the SMDFP, CGH [61], and GHCGP [62] methods.

Problem 4		SMDFP				CGH				GHCGP			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
1000	x_1	1	3	0.005096	0	-	-	-	-	25	75	0.074388	5.14×10^{-9}
	x_2	1	3	0.005338	0	1	3	0.003663	0	26	78	0.046803	4.09×10^{-9}
	x_3	1	3	0.005816	0	1	3	0.005606	0	26	79	0.04716	6.99×10^{-9}
	x_4	1	3	0.00575	0	1	3	0.005469	0	26	80	0.046384	7.42×10^{-9}
	x_5	1	3	0.005851	0	1	3	0.005245	0	26	81	0.047634	5.28×10^{-9}
	x_6	1	3	0.005918	0	1	3	0.005187	0	26	79	0.048149	4.88×10^{-9}
	x_7	1	3	0.004853	0	1	3	0.005062	0	23	70	0.042655	7.88×10^{-9}
	x_8	1	3	0.005734	0	1	3	0.005934	0	26	84	0.051013	7.34×10^{-9}
5000	x_1	1	3	0.038406	0	-	-	-	-	26	78	2.150775	4.61×10^{-9}
	x_2	1	3	0.026129	0	1	3	0.021583	0	26	78	1.483384	9.18×10^{-9}
	x_3	1	3	0.019968	0	1	3	0.017991	0	27	82	2.751638	6.28×10^{-9}
	x_4	1	3	0.030478	0	1	3	0.027568	0	27	83	2.081753	6.66×10^{-9}
	x_5	1	3	0.034382	0	1	3	0.023907	0	27	84	2.040673	4.75×10^{-9}
	x_6	1	3	0.019777	0	1	3	0.025307	0	27	82	2.46602	4.37×10^{-9}
	x_7	1	3	0.035994	0	1	3	0.040065	0	24	73	2.228755	7.05×10^{-9}
	x_8	1	3	0.031355	0	1	3	0.039022	0	27	87	1.830332	6.58×10^{-9}
10,000	x_1	1	3	0.028243	0	-	-	-	-	26	78	1.677716	6.53×10^{-9}
	x_2	1	3	0.02932	0	1	3	0.021768	0	27	81	2.064659	5.2×10^{-9}
	x_3	1	3	0.033628	0	1	3	0.043117	0	27	82	2.812976	8.88×10^{-9}
	x_4	1	3	0.029182	0	1	3	0.027671	0	27	83	2.747761	9.42×10^{-9}
	x_5	1	3	0.05026	0	1	3	0.023297	0	27	84	1.791871	6.72×10^{-9}
	x_6	1	3	0.029588	0	1	3	0.017539	0	27	82	2.440435	6.18×10^{-9}
	x_7	1	3	0.031805	0	1	3	0.032294	0	24	73	1.648455	9.97×10^{-9}
	x_8	1	3	0.038152	0	1	3	0.02327	0	27	87	1.710434	9.31×10^{-9}
50,000	x_1	1	3	0.078573	0	-	-	-	-	27	81	2.742683	5.84×10^{-9}
	x_2	1	3	0.103523	0	1	3	0.052539	0	28	84	3.509985	4.65×10^{-9}
	x_3	1	3	0.096328	0	1	3	0.060336	0	28	85	3.07001	7.95×10^{-9}
	x_4	1	3	0.087426	0	1	3	0.055083	0	28	86	2.848744	8.43×10^{-9}
	x_5	1	3	0.094959	0	1	3	0.059838	0	28	87	3.496331	6.01×10^{-9}
	x_6	1	3	0.092829	0	1	3	0.054967	0	28	85	3.266481	5.53×10^{-9}
	x_7	1	3	0.092697	0	1	3	0.063267	0	25	76	2.269198	8.92×10^{-9}
	x_8	1	3	0.096316	0	1	3	0.074379	0	28	90	2.783691	8.33×10^{-9}

Table 4. *Cont.*

Problem 4			SMDFP			CGH				GHCGP			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
100,000	x_1	1	3	0.150145	0	-	-	-	-	27	81	4.25115	8.26×10^{-9}
	x_2	1	3	0.1847	0	1	3	0.084738	0	28	84	4.937926	6.58×10^{-9}
	x_3	1	3	0.174993	0	1	3	0.096997	0	29	88	4.278757	4.5×10^{-9}
	x_4	1	3	0.248637	0	1	3	0.124909	0	29	89	4.49134	4.77×10^{-9}
	x_5	1	3	0.178261	0	1	3	0.126066	0	28	87	4.322899	8.5×10^{-9}
	x_6	1	3	0.155795	0	1	3	0.125207	0	28	85	4.35602	7.81×10^{-9}
	x_7	1	3	0.153226	0	1	3	0.120531	0	26	79	4.062214	5.04×10^{-9}
	x_8	1	3	0.182843	0	1	3	0.288392	0	29	93	4.299852	4.71×10^{-9}

Table 5. Numerical comparison of the SMDFP, CGH [61], and GHCGP [62] methods.

Problem 5			SMDFP			CGH				GHCGP			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
1000	x_1	1	4	0.004742	0	1	3	0.095815	0	15	61	0.019106	5.85×10^{-9}
	x_2	1	10	0.006129	0	1	3	0.004263	0	1	3	0.004059	0
	x_3	1	14	0.007935	0	1	7	0.004939	0	1	3	0.004083	0
	x_4	1	14	0.007325	0	1	12	0.006223	0	1	3	0.004035	0
	x_5	1	14	0.007355	0	1	14	0.006378	0	1	3	0.003865	0
	x_6	1	3	0.004434	0	1	3	0.004388	0	15	61	0.018663	3.98×10^{-9}
	x_7	1	3	0.004	0	1	3	0.004763	0	14	57	0.017273	4.93×10^{-9}
	x_8	1	14	0.007555	0	1	14	0.006207	0	1	3	0.005359	0
5000	x_1	1	4	0.037256	0	1	3	0.030271	0	16	65	0.675961	2.62×10^{-9}
	x_2	1	10	0.082444	0	1	3	0.0317	0	1	3	0.022237	0
	x_3	1	14	0.084766	0	1	7	0.071499	0	1	3	0.022972	0
	x_4	1	14	0.10313	0	1	12	0.104274	0	1	3	0.026511	0
	x_5	1	3	0.031836	0	1	14	0.106067	0	1	3	0.048799	0
	x_6	1	3	0.034784	0	1	3	0.025469	0	15	61	0.887118	8.9×10^{-9}
	x_7	1	3	0.049976	0	1	3	0.030696	0	15	61	1.163482	2.2×10^{-9}
	x_8	1	3	0.039773	0	1	14	0.100821	0	1	3	0.025777	0
10,000	x_1	1	4	0.054014	0	1	3	0.397306	0	16	65	1.078513	3.7×10^{-9}
	x_2	1	10	0.099402	0	1	3	0.400362	0	1	3	0.029786	0
	x_3	1	14	0.133006	0	1	7	0.096216	0	1	3	0.042265	0
	x_4	1	14	0.114411	0	1	12	0.148145	0	1	3	0.040303	0
	x_5	1	3	0.033535	0	1	14	0.248653	0	1	3	0.031149	0
	x_6	1	3	0.030994	0	1	3	0.24238	0	16	65	1.292146	2.52×10^{-9}
	x_7	1	3	0.026189	0	1	3	0.059157	0	15	61	1.200833	3.12×10^{-9}
	x_8	1	3	0.02614	0	1	14	0.139992	0	1	3	0.043985	0
50,000	x_1	1	4	0.071046	0	1	3	0.099884	0	16	65	1.572604	8.28×10^{-9}
	x_2	1	10	0.169775	0	1	3	0.083629	0	1	3	0.090689	0
	x_3	1	14	0.227858	0	1	7	0.187958	0	1	3	0.099092	0
	x_4	1	3	0.081543	0	1	12	0.510994	0	1	3	0.3273	0
	x_5	1	3	0.065031	0	1	14	0.193226	0	1	3	0.067268	0
	x_6	1	3	0.094322	0	1	3	0.077699	0	16	65	3.255643	5.63×10^{-9}
	x_7	1	3	0.060192	0	1	3	0.06505	0	15	61	2.191639	6.97×10^{-9}
	x_8	1	3	0.059951	0	1	14	0.498453	0	1	3	0.136817	0
100,000	x_1	1	4	0.136362	0	1	3	0.112411	0	17	69	3.476377	2.34×10^{-9}
	x_2	1	10	0.33991	0	1	3	0.484194	0	1	3	0.116341	0
	x_3	1	3	0.120604	0	1	7	0.237996	0	1	3	0.10336	0
	x_4	1	3	0.123234	0	1	12	0.323803	0	1	3	0.299915	0
	x_5	1	3	0.105494	0	1	14	0.291497	0	1	3	0.107072	0
	x_6	1	3	0.10725	0	1	3	0.1356	0	16	65	2.42926	7.96×10^{-9}
	x_7	1	3	0.095151	0	1	3	0.412831	0	15	61	1.704192	9.85×10^{-9}
	x_8	1	3	0.123426	0	1	14	0.293479	0	1	3	0.084475	0

Table 6. Numerical comparison of the SMDFP, CGH [61], and GHCGP [62] methods.

Problem 6			SMDFP			CGHM				GHCGPM			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
1000	x_1	28	365	0.177863	6.4×10^{-9}	101	1314	0.509568	8.63×10^{-9}	33	67	0.034528	8.84×10^{-9}
	x_2	30	391	0.18942	5.21×10^{-9}	101	1314	0.390828	9.95×10^{-9}	35	71	0.036322	8.9×10^{-9}
	x_3	31	404	0.193811	6.75×10^{-9}	101	1314	0.412585	9.85×10^{-9}	37	75	0.037483	6.28×10^{-9}
	x_4	33	430	0.212529	7.34×10^{-9}	100	1301	0.433422	9.8×10^{-9}	39	79	0.039543	8.22×10^{-9}
	x_5	36	469	0.229279	5.1×10^{-9}	93	1210	0.365524	8.57×10^{-9}	42	85	0.043653	7.71×10^{-9}
	x_6	27	352	0.174778	4.76×10^{-9}	99	1288	0.390179	9.68×10^{-9}	32	65	0.03258	5.96×10^{-9}
	x_7	24	313	0.161837	7.74×10^{-9}	94	1223	0.368178	8.53×10^{-9}	29	59	0.029921	6.68×10^{-9}
	x_8	37	482	0.24653	7.75×10^{-9}	3	40	0.014419	0	44	89	0.045735	6.28×10^{-9}

Table 6. *Cont.*

Problem 6		SMDFP				CGH				GHCGP			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
5000	x_1	29	378	2.980886	6.05×10^{-9}	105	1366	16.49603	8.52×10^{-9}	34	69	2.016212	9.63×10^{-9}
	x_2	31	404	3.319602	4.92×10^{-9}	105	1366	9.604319	9.83×10^{-9}	36	73	1.871193	9.68×10^{-9}
	x_3	32	417	2.993108	6.36×10^{-9}	105	1366	6.839324	9.74×10^{-9}	38	77	2.428632	6.81×10^{-9}
	x_4	34	443	2.798318	6.89×10^{-9}	104	1353	4.096109	9.74×10^{-9}	40	81	3.588929	8.89×10^{-9}
	x_5	37	482	2.724222	4.76×10^{-9}	97	1262	2.361667	9.43×10^{-9}	43	87	3.421294	8.29×10^{-9}
	x_6	28	365	1.821698	4.51×10^{-9}	103	1340	2.711408	9.55×10^{-9}	33	67	2.189047	6.5×10^{-9}
	x_7	25	326	1.464441	7.35×10^{-9}	98	1275	2.886032	8.4×10^{-9}	30	61	1.821143	7.3×10^{-9}
	x_8	38	495	1.908881	7.2×10^{-9}	3	40	0.079459	0	45	91	2.789199	6.72×10^{-9}
10,000	x_1	29	378	1.898784	8.53×10^{-9}	106	1379	4.306217	9.8×10^{-9}	35	71	2.232917	6.79×10^{-9}
	x_2	31	404	2.069526	6.93×10^{-9}	107	1392	4.291552	9.18×10^{-9}	37	75	2.195953	6.82×10^{-9}
	x_3	32	417	1.831673	8.96×10^{-9}	107	1392	4.323227	9.1×10^{-9}	38	77	3.49647	9.6×10^{-9}
	x_4	34	443	1.879557	9.69×10^{-9}	106	1379	3.935484	9.1×10^{-9}	41	83	2.109302	6.26×10^{-9}
	x_5	37	482	2.073277	6.69×10^{-9}	99	1288	3.094513	8.92×10^{-9}	44	89	2.172212	5.83×10^{-9}
	x_6	28	365	1.65292	6.35×10^{-9}	105	1366	3.4379	8.91×10^{-9}	33	67	1.488057	9.17×10^{-9}
	x_7	26	339	1.462072	4.51×10^{-9}	99	1288	3.260719	9.66×10^{-9}	31	63	2.113036	5.15×10^{-9}
	x_8	39	508	2.136944	4.4×10^{-9}	3	40	0.098955	0	45	91	2.322794	9.46×10^{-9}
50,000	x_1	30	391	5.510234	8.28×10^{-9}	110	1431	12.02104	9.53×10^{-9}	36	73	2.310523	7.57×10^{-9}
	x_2	32	417	5.422092	6.72×10^{-9}	111	1444	11.20052	8.93×10^{-9}	38	77	2.429677	7.6×10^{-9}
	x_3	33	430	5.211846	8.69×10^{-9}	111	1444	10.38166	8.85×10^{-9}	40	81	2.63147	5.35×10^{-9}
	x_4	35	456	5.418886	9.4×10^{-9}	110	1431	9.864932	8.86×10^{-9}	42	85	2.592365	6.97×10^{-9}
	x_5	38	495	5.911243	6.48×10^{-9}	103	1340	9.243294	8.77×10^{-9}	45	91	2.887271	6.5×10^{-9}
	x_6	29	378	4.547813	6.16×10^{-9}	109	1418	9.68457	8.67×10^{-9}	35	71	3.069208	5.11×10^{-9}
	x_7	27	352	4.509685	4.38×10^{-9}	103	1340	8.983611	9.39×10^{-9}	32	65	2.150055	5.74×10^{-9}
	x_8	39	508	6.3106	9.8×10^{-9}	3	40	0.315881	0	47	95	3.4158	5.26×10^{-9}
100,000	x_1	31	404	9.124879	5.09×10^{-9}	112	1457	19.99888	8.88×10^{-9}	37	75	3.532723	5.35×10^{-9}
	x_2	32	417	9.267631	9.5×10^{-9}	113	1470	18.60411	8.32×10^{-9}	39	79	3.615499	5.37×10^{-9}
	x_3	34	443	9.769236	5.34×10^{-9}	113	1470	19.53358	8.25×10^{-9}	40	81	3.712092	7.56×10^{-9}
	x_4	36	469	10.3297	5.78×10^{-9}	112	1457	21.29243	8.25×10^{-9}	42	85	4.354342	9.86×10^{-9}
	x_5	38	495	10.86344	9.16×10^{-9}	105	1366	20.03786	8.18×10^{-9}	45	91	4.165578	9.18×10^{-9}
	x_6	29	378	8.410524	8.71×10^{-9}	110	1431	18.72643	9.95×10^{-9}	35	71	3.300663	7.23×10^{-9}
	x_7	27	352	7.737803	6.19×10^{-9}	105	1366	16.81034	8.76×10^{-9}	32	65	3.313093	8.12×10^{-9}
	x_8	40	521	11.275	6.03×10^{-9}	3	40	0.561181	0	47	95	4.430573	7.44×10^{-9}

Table 7. Numerical comparison of the SMDFP, CGH [61], and GHCGP [62] methods.

Problem 7		SMDFP				CGH				GHCGP			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
1000	x_1	1	3	0.004846	0	108	1405	0.349376	8.21×10^{-9}	32	65	0.030379	5.96×10^{-9}
	x_2	1	3	0.004706	0	119	1548	0.356497	9.04×10^{-9}	32	65	0.030191	6.4×10^{-9}
	x_3	1	3	0.005136	0	-	-	-	-	29	59	0.028313	8.3×10^{-9}
	x_4	1	3	0.0051	0	1	3	0.003102	0	31	64	0.028523	9.58×10^{-9}
	x_5	1	3	0.004344	0	-	-	-	-	32	65	0.030716	6.9×10^{-9}
	x_6	1	3	0.00465	0	103	1340	0.339387	8.52×10^{-9}	31	63	0.028804	6.97×10^{-9}
	x_7	1	3	0.004371	0	95	1236	0.277541	8.16×10^{-9}	29	59	0.027149	5.88×10^{-9}
	x_8	1	3	0.005142	0	1	3	0.004269	0	35	74	0.03332	6.36×10^{-9}
5000	x_1	1	3	0.020114	0	111	1444	4.909827	9.82×10^{-9}	33	67	1.929645	6.66×10^{-9}
	x_2	1	3	0.013025	0	123	1600	6.337627	8.77×10^{-9}	33	67	2.086561	7.15×10^{-9}
	x_3	1	3	0.02639	0	-	-	-	-	30	61	1.893546	9.28×10^{-9}
	x_4	1	3	0.016253	0	1	3	0.010931	0	33	68	1.390724	5.36×10^{-9}
	x_5	1	3	0.026679	0	-	-	-	-	33	67	2.23439	7.72×10^{-9}
	x_6	1	3	0.027297	0	107	1392	1.259304	8.27×10^{-9}	32	65	1.953112	7.79×10^{-9}
	x_7	1	3	0.031613	0	98	1275	1.840177	9.76×10^{-9}	30	61	1.349774	6.57×10^{-9}
	x_8	1	3	0.036752	0	1	3	0.009336	0	36	76	2.060581	7.11×10^{-9}
10,000	x_1	1	3	0.031447	0	113	1470	3.136975	9.15×10^{-9}	33	67	1.818738	9.42×10^{-9}
	x_2	1	3	0.033583	0	125	1626	3.851012	8.18×10^{-9}	34	69	1.954061	5.06×10^{-9}
	x_3	1	3	0.024303	0	-	-	-	-	31	63	2.357244	6.56×10^{-9}
	x_4	1	3	0.052224	0	1	3	0.011473	0	33	68	3.237999	7.57×10^{-9}
	x_5	1	3	0.046682	0	-	-	-	-	34	69	1.586711	5.46×10^{-9}
	x_6	1	3	0.046979	0	108	1405	2.024726	9.49×10^{-9}	33	67	1.704647	5.51×10^{-9}
	x_7	1	3	0.057642	0	100	1301	3.125811	9.09×10^{-9}	30	61	1.669927	9.29×10^{-9}
	x_8	1	3	0.025922	0	1	3	0.011375	0	37	78	2.068357	5.03×10^{-9}

Table 7. *Cont.*

Problem 7		SMDFP				CGH				GHCGP			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
50,000	x_1	1	3	0.064974	0	117	1522	10.18713	8.89×10^{-9}	35	71	2.75523	5.27×10^{-9}
	x_2	1	3	0.061701	0	128	1665	10.09225	9.78×10^{-9}	35	71	2.387113	5.66×10^{-9}
	x_3	1	3	0.064387	0	-	-	-	-	32	65	2.055857	7.34×10^{-9}
	x_4	1	3	0.067186	0	1	3	0.019284	0	34	70	1.99498	8.47×10^{-9}
	x_5	1	3	0.089911	0	-	-	-	-	35	71	2.415085	6.1×10^{-9}
	x_6	1	3	0.059953	0	112	1457	5.988195	9.21×10^{-9}	34	69	2.627024	6.16×10^{-9}
	x_7	1	3	0.059474	0	104	1353	7.026984	8.83×10^{-9}	32	65	2.161008	5.19×10^{-9}
	x_8	1	3	0.102486	0	1	3	0.028383	0	38	80	2.533343	5.62×10^{-9}
100,000	x_1	1	3	0.150496	0	119	1548	18.04442	8.28×10^{-9}	35	71	3.709317	7.45×10^{-9}
	x_2	1	3	0.119297	0	130	1691	17.32583	9.11×10^{-9}	35	71	4.106231	8×10^{-9}
	x_3	1	3	0.120414	0	-	-	-	-	33	67	3.193129	5.19×10^{-9}
	x_4	1	3	0.167358	0	1	3	0.055669	0	35	72	3.486973	5.99×10^{-9}
	x_5	1	3	0.14736	0	-	-	-	-	35	71	3.104115	8.63×10^{-9}
	x_6	1	3	0.123877	0	114	1483	12.50694	8.59×10^{-9}	34	69	3.558606	$8.71\text{E}\times10^{-9}$
	x_7	1	3	0.113269	0	106	1379	14.34735	8.22×10^{-9}	32	65	3.108505	7.35×10^{-9}
	x_8	1	3	0.111948	0	1	3	0.046828	0	38	80	3.532805	7.95×10^{-9}

Table 8. Numerical comparison of the SMDFP, CGH [61], and GHCGP [62] methods.

Problem 8		SMDFP				CGH				GHCGP			
DIMENSION	**INITIAL POINT**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**	**ITER**	**FEV**	**CPUT**	**NORM**
1000	x_1	2	5	0.006257	0	7	26	0.064518	0	-	-	-	-
	x_2	1	3	0.004575	0	9	30	0.013856	0	50	108	0.060925	0
	x_3	2	5	0.005677	0	6	24	0.011035	0	-	-	-	-
	x_4	2	5	0.006564	0	131	1671	0.415251	8.78×10^{-9}	-	-	-	-
	x_5	2	5	0.005898	0	7	26	0.012663	0	4	17	0.007869	0
	x_6	2	5	0.005661	0	7	26	0.013313	0	-	-	-	-
	x_7	2	5	0.006256	0	135	1679	0.398362	8.98×10^{-9}	-	-	-	-
	x_8	3	7	0.006335	0	5	11	0.00916	0	-	-	-	-
5000	x_1	2	5	0.071744	0	7	26	0.278915	0	-	-	-	-
	x_2	1	3	0.032757	0	9	30	0.335638	0	19	46	0.321704	6.24×10^{-9}
	x_3	2	5	0.074475	0	6	24	0.227703	0	-	-	-	-
	x_4	2	5	0.150227	0	139	1786	10.12413	9.92×10^{-9}	17	42	0.140003	7.16×10^{-9}
	x_5	2	5	0.054634	0	5	22	0.309578	0	5	20	0.054782	0
	x_6	2	5	0.102503	0	7	26	0.290641	0	-	-	-	-
	x_7	2	5	0.067849	0	129	1623	9.42968	8.65×10^{-9}	-	-	-	-
	x_8	2	5	0.12588	0	4	9	0.179896	0	6	26	0.033651	0
10,000	x_1	2	5	0.109853	0	7	26	0.429911	0	-	-	-	-
	x_2	1	3	0.064967	0	9	30	0.433819	0	18	44	0.113456	6.24×10^{-9}
	x_3	2	5	0.109943	0	6	24	0.249119	0	-	-	-	-
	x_4	2	5	0.101969	0	140	1799	11.30977	9.87×10^{-9}	16	40	0.094243	7.16×10^{-9}
	x_5	2	5	0.066711	0	5	22	0.339357	0	5	20	0.047766	0
	x_6	2	5	0.069338	0	7	26	0.247131	0	-	-	-	-
	x_7	2	5	0.086838	0	9	30	0.300432	0	-	-	-	-
	x_8	2	5	0.083051	0	4	9	0.115857	0	7	29	0.056413	0
50,000	x_1	2	5	0.131377	0	7	26	3.688616	0	-	-	-	-
	x_2	1	3	0.070502	0	9	30	0.623855	0	15	38	0.261414	9.98×10^{-9}
	x_3	2	5	0.257994	0	6	24	0.577014	0	-	-	-	-
	x_4	2	5	0.15874	0	135	1734	17.44768	9.73×10^{-9}	14	36	0.241642	5.73×10^{-9}
	x_5	2	5	0.144073	0	5	22	0.37145	0	6	23	0.157215	0
	x_6	2	5	0.128614	0	7	26	0.426915	0	-	-	-	-
	x_7	2	5	0.152141	0	9	30	3.662605	0	-	-	-	-
	x_8	2	5	0.171881	0	5	22	3.374986	0	7	29	0.172393	0
100,000	x_1	2	5	0.383221	0	7	26	12.88737	0	-	-	-	-
	x_2	1	3	0.256944	0	9	30	0.769409	0	14	36	0.546488	9.98×10^{-9}
	x_3	2	5	0.231476	0	6	24	0.52902	0	-	-	-	-
	x_4	2	5	0.327778	0	133	1708	22.37772	8.54×10^{-9}	13	34	0.497945	5.73×10^{-9}
	x_5	2	5	0.25519	0	5	22	0.330482	0	6	23	0.358825	0
	x_6	2	5	0.312419	0	7	26	0.487292	0	-	-	-	-
	x_7	2	5	0.228052	0	9	30	11.78943	0	-	-	-	-
	x_8	2	5	0.320283	0	5	22	7.510984	0	7	29	0.39418	0

Next, the robustness of the SMDFP algorithm is illustrated by Figures 1–3 showing the performance profiles based on Dolan and Moré procedure [69]. In Figure 1, the top curve showed the SMDFP algorithm with respect to the number of iterations, which leads the remaining curves of CGH [61] and GHCGP methods [62]. Similarly, in the case of function evaluations and CPU times, the SMDFP algorithm has better performance, as

shown in Figures 2 and 3, respectively. All of the above factors clearly showed that our proposed algorithm is leading and the best in all aspects, i.e., in terms of iterations, function evaluations, and CPU times.

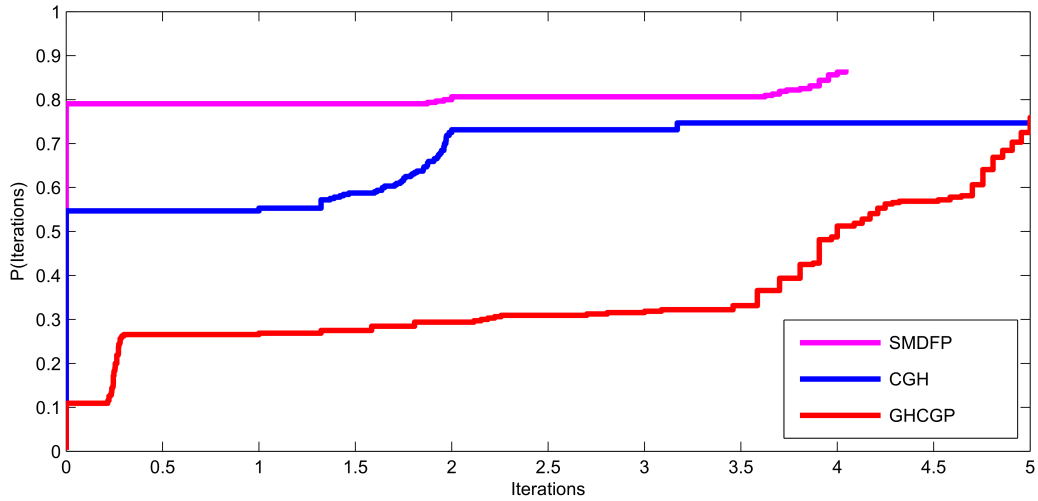

Figure 1. Performance of the SMDFP, CGH [61], and GHCGP [62] methods for the number of iterations.

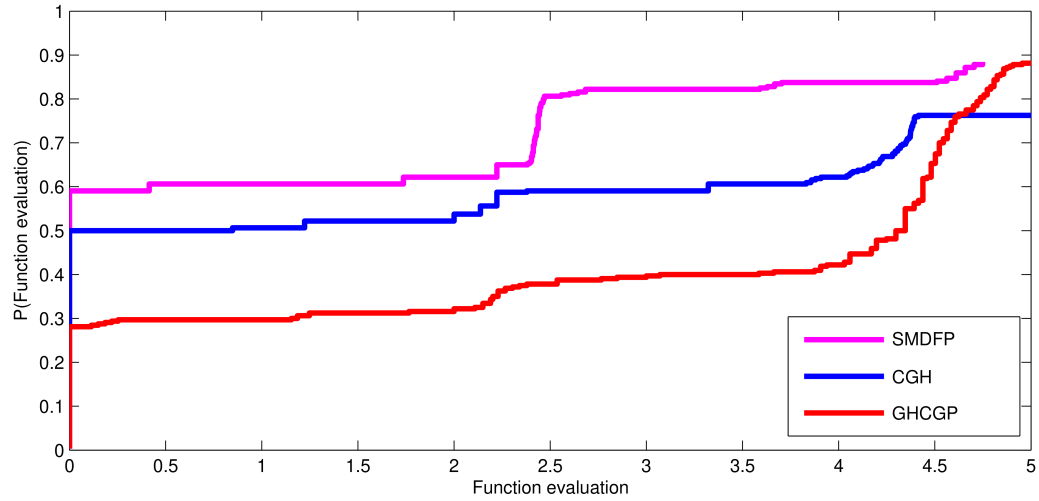

Figure 2. Performance of the SMDFP, CGH [61], and GHCGP [62] methods for the number of function evaluations.

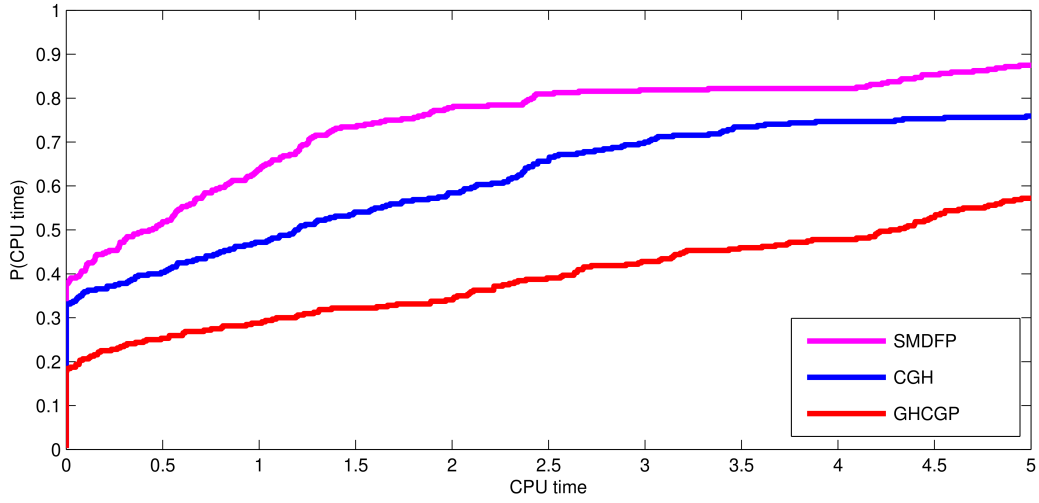

Figure 3. Performance of the SMDFP, CGH [61], and GHCGP [62] methods for the number of CPU time.

5. Applications

5.1. Image Restoration

This section describes techniques for reducing noise and recovering lost image resolutions. Its applications are in the field of medical imaging [70], astronomical imaging [71], file restoration [72], and image coding [73]. Let x^* be the original sparse signal, and $t \in R^m$ be an observation satisfying

$$t = Ux^*, \tag{61}$$

where $U \in R^{m \times n}(m << n)$ is a linear operator. This problem is used for finding solutions to the sparse ill-conditioned linear system of equations. According to Bruckstein et al. [74], a function containing a quadratic (l_2) error term as well as a sparse l_1–regularization term is minimized as,

$$\min_x \left\{ \frac{1}{2}\|t - Ux\|_2^2 + \beta\|x\|_1 \right\}, \tag{62}$$

where $x \in R^n$, and β is a nonnegative balance parameter, $\|x\|_2$ denotes the Euclidean norm of x, and $\|x\|_1$ is the l_1–norm of x. Problem (62) is a convex unconstrained minimization problem commonly used in compressive sensing if the original signal is sparse or near to sparse on some orthogonal basis.

Many iterative techniques have been introduced in literature such as Figueiredo et al. [75], Hale et al. [76], Figueiredo et al. [77], Van Den Berg et al. [78], Beck et al. [79], and Hager et al. [80] for finding a solution to (62). The GPRS method has the following steps to present (62) as a quadratic problem.

Let $x \in \mathbb{R}^n$ be categorized into two different classes, namely, its positive and negative parts:

$$x = q - r, \quad q \geq 0, \quad r \geq 0, \tag{63}$$

where $q_i = (x_i)_+, r_i = (-x_i)_+$ for all $i = 1, 2, \ldots, n$, and $(.)_+ = \max\{0, .\}$. Since, by definition of ℓ_1–norm, we have $\|x\|_1 = e_n^T q + e_n^T r$, where $e_n = (1, 1, \ldots, 1)^T \in R^n$. Thus, (62) can be written as:

$$\min_{q,r} \frac{1}{2}\|t - U(q - r)\|_2^2 + \beta e_n^T q + \beta e_n^T r, \quad q \geq 0, \quad r \geq 0, \tag{64}$$

which is a bound–constrained and quadratic problem. Moreover, following Figueiredo et al. [77], a standard form of problem (64) can be written as:

$$\min_v \frac{1}{2} v^T A v + c^T v, \quad \text{such that} \quad v \geq 0, \tag{65}$$

where $v = \begin{pmatrix} q \\ r \end{pmatrix}, b = U^T t, c = \beta e_{2n} + \begin{pmatrix} -b \\ b \end{pmatrix}$, and $A = \begin{pmatrix} U^T U & -U^T U \\ -U^T U & U^T U \end{pmatrix}$. Problem (65) is convex quadratic because A is a positive semi-definite matrix as proved by Xiao et al. [5]. He also transformed (65) into a linear variable inequality problem, which is equivalent to a linear complementarity problem. However, v is a solution to the linear complementarity problem if and only if it is a solution to the nonlinear equation

$$f(v) = \min\{v, Av + c\} = 0, \tag{66}$$

which is continuously monotone, as shown by Xiao et al. [5]. So, problem (66) can be solved using the SMDFP method.

5.2. Implementation

Table 9 shows the numerical comparison for four different methods, namely; SMDFP method, CGD method [45], PSGM method [81], and TPRP method [82] by applying on

seven different images, namely, baby, COMSATS, fox, horse, Lena, and Thai-Culture. The mean of squared error (MSE) [45] between the blurred x^* and restored x images is

$$\text{MSE} = \frac{1}{n}\|x^* - x\|^2,$$

and the signal–to–ratio (SNR) [45] of the recovered images is

$$\text{SNR} = 20 \times \log_{10}\left(\frac{\|x^*\|}{\|x - x^*\|}\right).$$

By both MSE and SNR, we measure the image restoration quality by taking $\rho = 6$ and $\theta = 0.0001$. We studied a compressive sensing situation in which the aim was to reconstruct a length-n sparse signal from m observations, where $m << n$. We test a modest size signal with $n = 211$, $m = 29$, and the original has 26 randomly non-zero elements due to the PC's storage restrictions. The random U is the Gaussian matrix created by the Matlab command $randn(m, n)$. The measurement t in this test involves noise,

$$t = Ux^* + \phi, \tag{67}$$

where ϕ is the Gaussian noise distributed as $N(0, \sigma^2 I)$ with $\sigma^2 = 10^{-4}$. We used $f(x) = \frac{1}{2}\|t - Ux\|_2^2 + \beta\|x\|_1$ as the merit function, $x_0 = U^T t$, β is compelled to decrease as adopted in [75], and the iteration terminates if

$$Tolerance = \frac{\|f(x_k) - f(x_{k-1})\|}{\|f(x_{k-1})\|} < 10^{-10}. \tag{68}$$

All of the codes in this section are also executed on the same machine and software as described in Section 4. In summary, the performance of the SMDFP method is better than CGD [45], PSGM [81], and TPRP [82] methods for the number of iterations, CPU time, and recovered images SNR quality as clear from Table 9. Consequently, it can be seen from Figures 4–10 that the proposed method has restored all the problems with fewer iterations and CPU time.

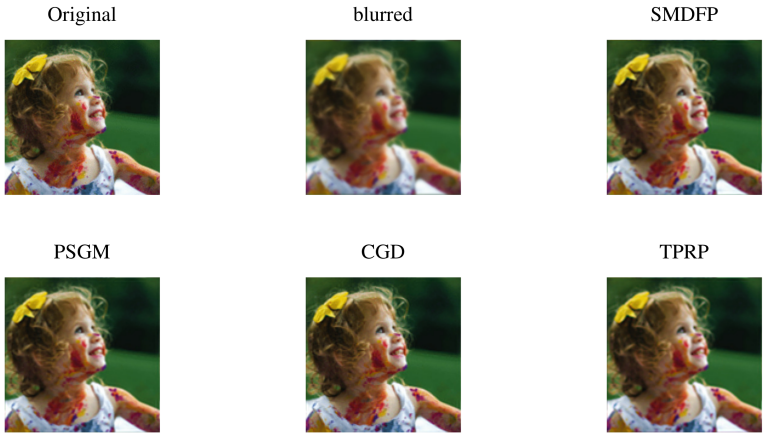

Figure 4. Comparison of the original image, blurred image, SMDFP image, CGD image, PSGM image, and TPRP image for the number of iterations, CPU time, MSE, and SNR.

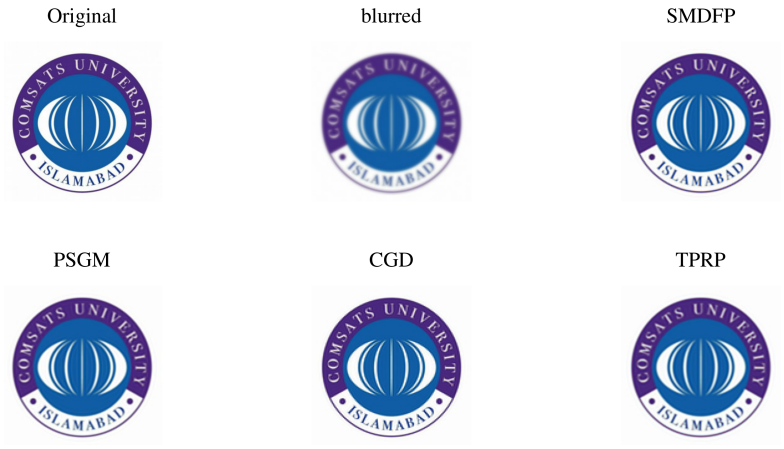

Figure 5. Comparison of the original image, blurred image, SMDFP image, CGD image, PSGM image, and TPRP image for the number of iterations, CPU time, MSE, and SNR.

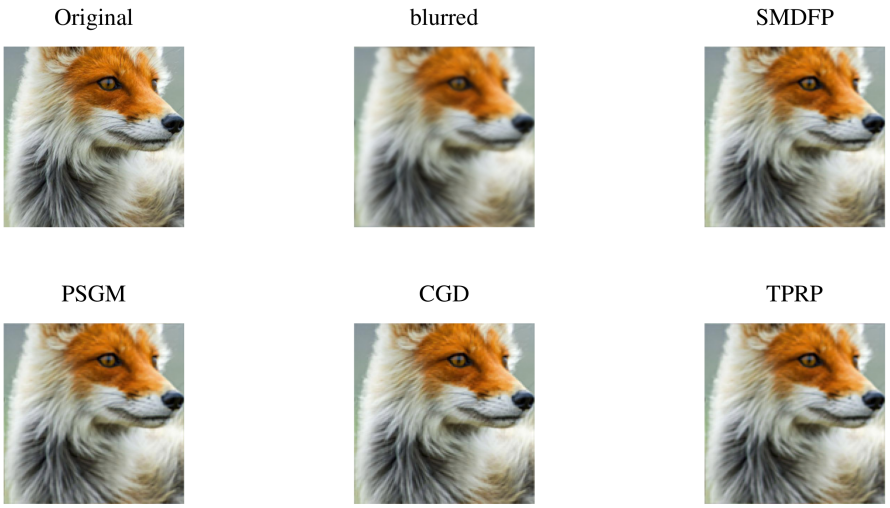

Figure 6. Comparison of the original image, blurred image, SMDFP image, CGD image, PSGM image, and TPRP image for the number of iterations, CPU time, MSE, and SNR.

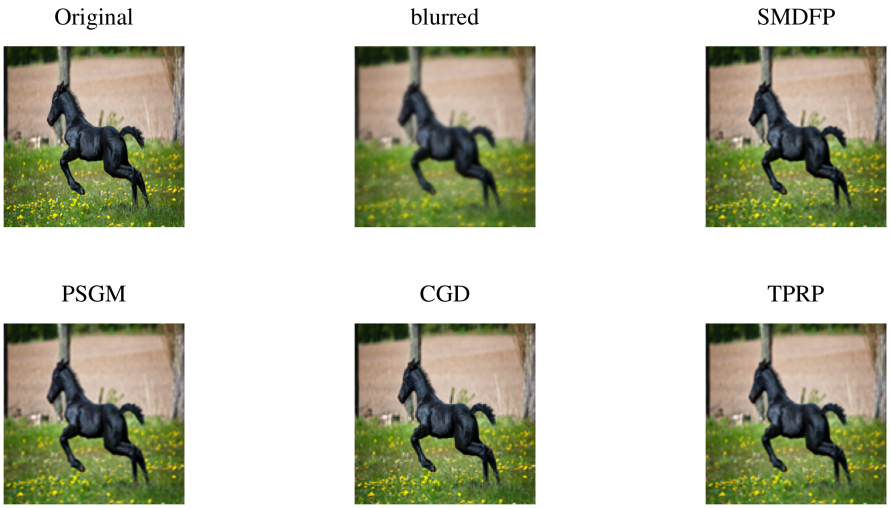

Figure 7. Comparison of the original image, blurred image, SMDFP image, CGD image, PSGM image, and TPRP image for the number of iterations, CPU time, MSE, and SNR.

Table 9. Numerical comparison of SMDFP, CGD [45], PSGM [81], and TPRP [82] methods.

	SMDFP					CGD					PSGM					TPRP				
Pictures	**ITER**	**CPUT**	**MSE**	**SNR**	**SSIM**	**ITER**	**CPUT**	**MSE**	**SNR**	**SSIM**	**ITER**	**CPUT**	**MSE**	**SNR**	**SSIM**	**ITER**	**CPUT**	**MSE**	**SNR**	**SSIM**
Baby	69	4.55	3.88×10^{1}	22.79	0.92	713	42.19	4.46×10^{1}	22.19	0.93	27	1.55	3.87×10^{1}	22.8	0.93	317	1366.58	4.56×10^{1}	22.09	0.92
COMSATS	14	0.64	1.83×10^{2}	23.9	0.87	671	49.02	1.69×10^{2}	24.24	0.86	23	1.88	1.52×10^{2}	24.71	0.87	50	1020.66	1.87×10^{2}	23.79	0.87
Fox	39	4.25	7.33×10^{1}	24.45	0.84	474	62.83	7.45×10^{1}	24.38	0.86	18	2.36	7.37×10^{1}	24.42	0.84	58	228.41	7.76×10^{1}	24.2	0.84
Horse	16	1.16	1.20×10^{2}	19.52	0.81	645	44.06	1.06×10^{2}	20.07	0.85	42	2.59	1.07×10^{2}	20	0.83	109	248.86	1.10×10^{2}	19.9	0.81
Lena	14	0.78	6.30×10^{1}	22.88	0.84	491	34.63	6.05×10^{1}	23.05	0.88	26	2.05	5.81×10^{1}	23.23	0.87	101	1175.6	5.91×10^{1}	23.16	0.84
Marwat	34	2.02	1.52×10^{2}	20.43	0.81	856	58.33	1.53×10^{2}	20.4	0.84	38	2.42	1.46×10^{2}	20.61	0.82	472	707.05	1.61×10^{2}	20.18	0.81
Thai fabric pattern in Phetchabun	12	0.97	2.46×10^{2}	17.69	0.66	565	48.23	2.47×10^{2}	17.67	0.7	22	1.84	2.38×10^{2}	17.83	0.68	156	1445.32	2.31×10^{2}	17.97	0.69

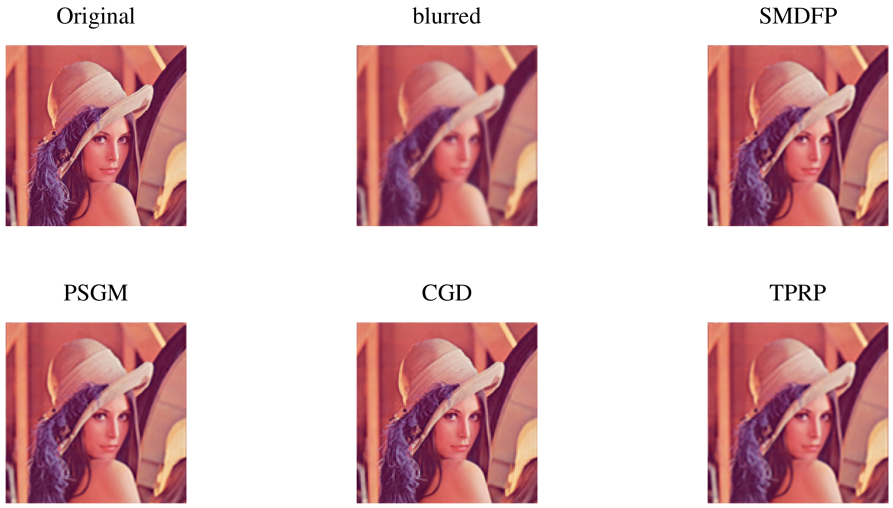

Figure 8. Comparison of the original image, blurred image, SMDFP image, CGD image, PSGM image, and TPRP image for the number of iterations, CPU time, MSE, and SNR.

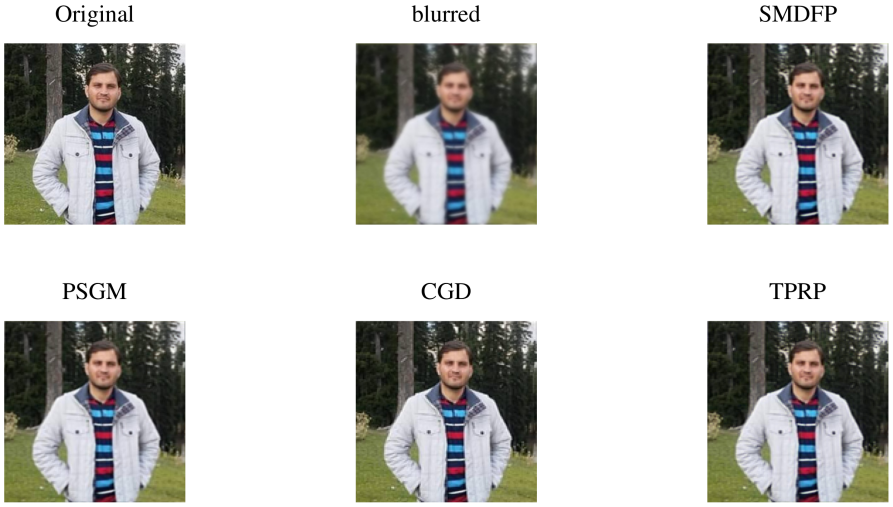

Figure 9. Comparison of the original image, blurred image, SMDFP image, CGD image, PSGM image, and TPRP image for the number of iterations, CPU time, MSE, and SNR.

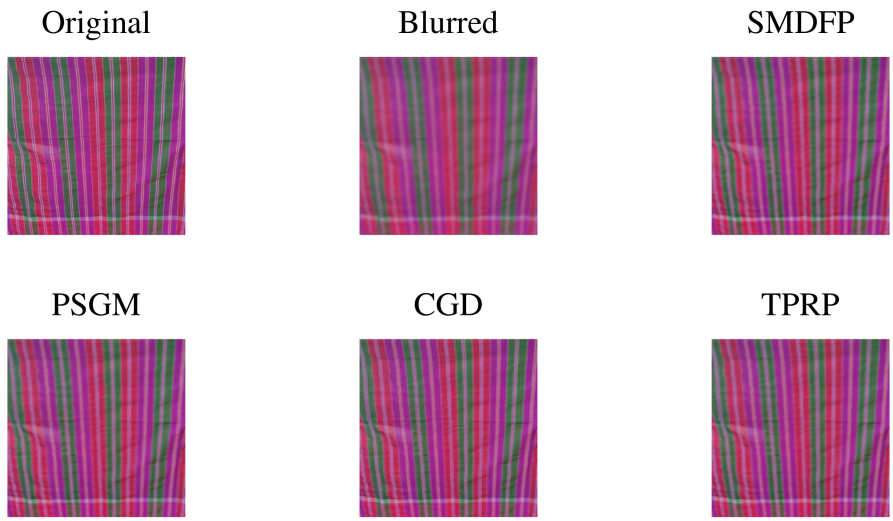

Figure 10. Comparison of the original image, blurred image, SMDFP image, CGD image, PSGM image, and TPRP image for the number of iterations, CPU time, MSE, and SNR.

6. Conclusions

This paper presents a one-parameter SMDFP method for solving a system of monotone nonlinear equations with convex constraints. We scaled one term of the DFP update formula and found the optimal value of the scaled parameter using the idea of measure function. Based on the new optimal value of the scaled parameter, we derived a modified search direction for the DFP algorithm. The proposed method is globally convergent using the monotonicity and Lipschitz continuous assumptions. The method's robustness is demonstrated by solving large-scale monotone nonlinear equations and making comparisons with the related CGH and GHCGP methods. Lastly, the algorithm is successfully implemented for solving some image restoration problems. The proposed scale DFP direction can further be applied to solve unconstrained optimization problems, motion control of two coplanar joint robot manipulators, and many more problems.

Author Contributions: Conceptualization, N.U., J.S., X.J., A.M.A., N.P. and B.P.; Methodology, N.U., A.S., X.J., A.M.A. and N.P.; Software, J.S., X.J. , A.M.A. and N.P.; Validation, J.S., X.J., N.P. and S.K.S.; Formal analysis, A.S.; Investigation, N.U., A.S. and S.K.S.; Resources, A.M.A.; Data curation, N.U. and S.K.S.; Writing—original draft, N.U., J.S., X.J., A.M.A., N.P. , S.K.S. and B.P. ; Writing—review & editing, N.U., A.S., N.P. and B.P.; Visualization, A.M.A. and N.P.; Supervision, A.S., J.S., X.J. and B.P.; Project administration, N.P. and B.P.; Funding acquisition, N.P. and B.P. All authors have read and agreed to the published version of the manuscript.

Funding: The sixth author was supported by Phetchabun Rajabhat University and Thailand Science Research and Innovation (grant number 182093). The eighth author was partially supported by Chiang Mai University and Fundamental Fund 2023, Chiang Mai University, and the NSRF via the Program Management Unit for Human Resources and Institutional Development, Research and Innovation (grant number B05F640183).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Prajna, S.; Parrilo, P.A.; Rantzer, A. Nonlinear control synthesis by convex optimization, *IEEE Trans. Autom. Control* **2004**, *49*, 310–314. [CrossRef]
2. Abubakar, A.B.; Kumam, P.; Mohammad, H.; Awwal, A.M. An efficient conjugate gradient method for convex constrained monotone nonlinear equations with applications. *Mathematics* **2019**, *7*, 767. [CrossRef]
3. Hu, Y.; Wang, Y. An efficient projected gradient method for convex constrained monotone equations with applications in compressive sensing. *J. Appl. Math. Phys.* **2020**, *8*, 983–998. [CrossRef]
4. Liu, J.K.; Du, X.L. A gradient projection method for the sparse signal reconstruction in compressive sensing. *Appl. Anal.* **2018**, *97*, 2122–2131. [CrossRef]
5. Xiao, Y.; Wang, Q.; Hu, Q. Non–smooth equations based method for (l_1)–norm problems with applications to compressive sensing, *Nonlinear Anal. Theory Methods Appl.* **2011**, *74*, 3570–3577. [CrossRef]
6. Luo, Z.Q.; Yu, W. An introduction to convex optimization for communications and signal processing. *IEEE J. Sel. Areas Commun.* **2006**, *24*, 1426–1438.
7. Evgeniou, T.; Pontil, M.; Toubia, O. A convex optimization approach to modelling consumer heterogeneity in conjoint estimation. *Mark. Sci.* **2007**, *26*, 805–818. [CrossRef]
8. Bello, L.; Raydan, M. Convex constrained optimization for the seismic reflection tomography problem. *J. Appl. Geophys.* **2007**, *62*, 158–166. [CrossRef]
9. Solodov, M.V.; Svaiter, B.F. A globally convergent inexact Newton method for systems of monotone equations. In *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*; Fukushima, M., Qi, L., Eds.; Springer: Boston, MA, USA, 1998; Volume 22, pp. 355–369.
10. Davidon, W.C. Variable metric method for minimization. *SIAM J. Optim.* **1991**, *1*, 1–17. [CrossRef]
11. Fletcher, R.; Powell, M.J.D. A rapidly convergent descent method for minimization. *Comput. J.* **1963**, *6*, 163–168. [CrossRef]
12. Dingguo, P. Superlinear convergence of the DFP algorithm without exact line search. *Acta Math. Appl. Sin.* **2001**, *17*, 430–432. [CrossRef]
13. Dingguo, P.; Weiwen, T. A class of Broyden algorithms with revised search directions. *Asia–Pac. J. Oper. Res.* **1997**, *14*, 93–109.
14. Pu, D. Convergence of the DFP algorithm without exact line search. *J. Optim. Theory Appl.* **2002**, *112*, 187–211. [CrossRef]
15. Pu, D.; Tian, W. The revised DFP algorithm without exact line search. *J. Comput. Appl. Math.* **2003**, *154*, 319–339. [CrossRef]

16. Kanzow, C.; Yamashita, N.; Fukushima, M. Levenberg–Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints. *J. Comput. Appl. Math.* **2005**, *173*, 321–343. [CrossRef]
17. Bellavia, S.; Macconi, M.; Morini, B. A scaled trust–region solver for constrained nonlinear equations. *Comput. Optim. Appl.* **2004**, *28*, 31–50. [CrossRef]
18. Bellavia, S.; Morini, B. An interior global method for nonlinear systems with simple bounds. *Optim. Methods Softw.* **2005**, *20*, 453–474. [CrossRef]
19. Bellavia, S.; Morini, B.; Pieraccini, S. Constrained Dogleg methods for nonlinear systems with simple bounds. *Comput. Optim. Appl.* **2012**, *53*, 771–794. [CrossRef]
20. Yu, G. A derivative–free method for solving large–scale nonlinear systems of equations. *J. Ind. Manag. Optim.* **2010**, *6*, 149–160. [CrossRef]
21. Liu, J.; Feng, Y. A derivative–free iterative method for nonlinear monotone equations with convex constraints. *Numer. Algorithms* **2019**, *82*, 245–262. [CrossRef]
22. Mohammad, H.; Abubakar, A.B. A descent derivative–free algorithm for nonlinear monotone equations with convex constraints. *RAIRO–Oper. Res.* **2020**, *54*, 489–505. [CrossRef]
23. Wang, C.; Wang, Y. A super–linearly convergent projection method for constrained systems of nonlinear equations. *J. Glob. Optim.* **2009**, *44*, 283–296. [CrossRef]
24. Ma, F.; Wang, C. Modified projection method for solving a system of monotone equations with convex constraints. *J. Appl. Math. Comput.* **2010**, *34*, 47–56. [CrossRef]
25. Yu, G.; Niu, S.; Ma, J. Multivariate spectral gradient projection method for nonlinear monotone equations with convex constraints. *J. Ind. Manag. Optim.* **2013**, *9*, 117–129. [CrossRef]
26. Liu, J.K.; Li, S.J. A projection method for convex constrained monotone nonlinear equations with applications. *Comput. Math. Appl.* **2015**, *70*, 2442–2453. [CrossRef]
27. Ou, Y.; Li, J. A new derivative–free SCG–type projection method for nonlinear monotone equations with convex constraints. *J. Appl Math Comput.* **2018**, *56*, 195–216. [CrossRef]
28. Liu, J.K.; Xu, J.L.; Zhang, L.Q. Partially symmetrical derivative–free Liu–Storey projection method for convex constrained equations. *Int. J. Comput. Math.* **2019**, *96*, 1787–1798. [CrossRef]
29. Zheng, L.; Yang, L.; Liang, Y. A modified spectral gradient projection method for solving non–linear monotone equations with convex constraints and its application. *IEEE Access.* **2020**, *8*, 92677–92686. [CrossRef]
30. Liu, Y.; Storey, C. Efficient generalized conjugate gradient algorithms, Part 1: Theory. *J. Optim. Theory Appl.* **1991**, *69*, 129–137. [CrossRef]
31. Dai, Y.H.; Yuan, Y. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM J. Optim.* **1999**, *10*, 177–182. [CrossRef]
32. Liu, S.Y.; Huang, Y.Y.; Jiao, H.W. Sufficient decent conjugate gradient methods for solving convex constrained nonlinear monotone equations. *Abstr. Appl. Anal.* **2014**, *2014*, 305643.
33. Sun, M.; Liu, J. New hybrid conjugate gradient projection method for the convex constrained equations. *Calcolo* **2016**, *53*, 399–411. [CrossRef]
34. Wang, X.Y.; Li, S.J.; Kou, X.P. A self–adaptive three–term conjugate gradient method for monotone nonlinear equations with convex constraints. *Calcolo* **2016**, *53*, 133–145. [CrossRef]
35. Gao, P.; He, C. An efficient three–term conjugate gradient method for nonlinear monotone equations with convex constraints. *Calcolo* **2018**, *55*, 53. [CrossRef]
36. Ibrahim, A.H.; Garba, A.I.; Usman, H.; Abubakar, J.; Abubakar, A.B. Derivative–free RMIL conjugate gradient method for convex constrained equations, *Thai J. Math.* **2019**, *18*, 212–232.
37. Abubakar, A.B.; Rilwan, J.; Yimer, S.E.; Ibrahim, A.H.; Ahmed, I. Spectral three–term conjugate descent method for solving nonlinear monotone equations with convex constraints. *Thai J. Math.* **2020**, *18*, 501–517.
38. Zhou, G.; Toh, K.C. Superlinear convergence of a Newton type algorithm for monotone equations. *J. Optim. Theory Appl.* **2005**, *125*, 205–221. [CrossRef]
39. Zhou, W.J.; Li, D.H. A globally convergent BFGS method for nonlinear monotone equations without any merit functions. *Math. Comput.* **2008**, *77*, 2231–2240. [CrossRef]
40. Zhou, W.; Li, D. Limited memory BFGS method for nonlinear monotone equations. *J. Comput. Math.* **2007**, *25*, 89–96.
41. Zhang, L.; Zhou, W. Spectral gradient projection method for solving nonlinear monotone equations. *J. Comput. Appl. Math.* **2006**, *196*, 478–484. [CrossRef]
42. Barzilai, J.; Borwein, J.M. Two–point step size gradient methods. *IMA J. Numer. Anal.* **1988**, *8*, 141–148. [CrossRef]
43. Wang, C.; Wang, Y.; Xu, C. A projection method for a system of nonlinear monotone equations with convex constraints. *Math. Methods Oper. Res.* **2007**, *66*, 33–46. [CrossRef]
44. Yu, Z.; Lin, J.; Sun, J.; Xiao, Y.; Liu, L.; Li, Z. Spectral gradient projection method for monotone nonlinear equations with convex constraints. *Appl. Numer. Math.* **2009**, *59*, 2416–2423. [CrossRef]
45. Xiao, Y.; Zhu, H. A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing. *J. Math. Anal. Appl.* **2013**, *405*, 310–319. [CrossRef]

46. Hager, W.W.; Zhang, H. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM J. Optim.* **2005**, *16*, 170–192. [CrossRef]
47. Hager, W.W.; Zhang, H. CG_DESCENT, a conjugate gradient method with guaranteed descent. *ACM Trans. Math. Softw.* **2006**, *32*, 113–137. [CrossRef]
48. Muhammed, A.A.; Kumam, P.; Abubakar, A.B.; Wakili, A.; Pakkaranang, N. A new hybrid spectral gradient projection method for monotone system of nonlinear equations with convex constraints.*Thai J. Math.* **2018**, *16*, 125–147.
49. Sabi'u, J.; Shah, A.; Waziri, M.Y.; Ahmed, K. Modified Hager–Zhang conjugate gradient methods via singular value analysis for solving monotone nonlinear equations with convex constraint.*Int. J. Comput. Methods* **2021**, *18*, 2050043. [CrossRef]
50. Sabi'u, J.; Shah, A.; Waziri, M.Y. A modified Hager–Zhang conjugate gradient method with optimal choices for solving monotone nonlinear equations. *Int. J. Comput. Math.* **2021**, *99*, 332–354. [CrossRef]
51. Sabi'u, J.; Shah, A. An efficient three–term conjugate gradient–type algorithm for monotone nonlinear equations. *RAIRO Oper. Res.* **2021**, *55*, 1113–1127. [CrossRef]
52. Andrei, N. A double parameter self–scaling memoryless BFGS method for unconstrained optimization. *Comput. Appl. Math.* **2020**, *39*, 1–14. [CrossRef]
53. Andrei, N. A note on memory–less SR1 and memory–less BFGS methods for large–scale unconstrained optimization. *Numer. Algorithms* **2021**, *99*, 223–240. [CrossRef]
54. Fletcher, R. *Practical Methods of Optimization*, 2nd ed.; John Wiley & Sons: New York, NY, USA, 1990.
55. Byrd, R.H.; Nocedal, J. A tool for the analysis of quasi–Newton methods with application to unconstrained minimization. *SIAM J. Numer. Anal.* **1989**, *26*, 727–739. [CrossRef]
56. Andrei, N. A double parameter scaled BFGS method for unconstrained optimization. *J. Comput. Appl. Math.* **2018**, *332*, 26–44. [CrossRef]
57. Fletcher, R. An overview of unconstrained optimization, In *The State of the Art*; Algorithms for Continuous Optimization; Spedicato E., Ed.; Kluwer Academic Publishers: Boston, MA, USA, 1994; pp. 109–143.
58. Sun, W.; Yuan, Y.X. *Optimization Theory and Methods, Nonlinear Programming*; Springer Science + Business Media: New York, NY, USA, 2006.
59. Behrens, R.T.; Scharf, L.L. Signal processing applications of oblique projection operators. *IEEE Trans. Signal Process* **1994**, *42*, 1413–1424. [CrossRef]
60. Zarantonello, E.H. Projections on convex sets in Hilbert space and spectral theory: Part I. Projections on convex sets: Part II. Spectral theory. *Contrib. Nonlinear Funct. Anal.* **1971**, *5*, 237–424.
61. Halilu, A.S.; Majumder, A.; Waziri, M.Y.; Ahmed, K. Signal recovery with convex constrained nonlinear monotone equations through conjugate gradient hybrid approach. *Math. Compu. Simul.* **2021**, *187*, 520–539. [CrossRef]
62. Yin, J.H.; Jian, J.B.; Jiang, X.Z. A generalized hybrid CGPM–based algorithm for solving large–scale convex constrained equations with applications to image restoration. *J. Comput. Appl. Math.* **2021**, *391*, 113423. [CrossRef]
63. Sabi'u, J.; Shah, A.; Waziri, M.Y. Two optimal Hager–Zhang conjugate gradient methods for solving monotone nonlinear equations. *Appl. Numer. Math.* **2020**, *153*, 217–233. [CrossRef]
64. Ullah, N.; Sabi'u, J.; Shah, A. A derivative–free scaling memoryless Broyden–Fletcher–Goldfarb–Shanno method for solving a system of monotone nonlinear equations. *Numeri. lin. alge. with appl.* **2021**, *28*, e2374. [CrossRef]
65. Abubakar, A.B.; Sabi'u, J.; Kumam, P.; Shah, A. Solving nonlinear monotone operator equations via modified SR1 update. *J. Appl. Math. Comput.* **2021**, *67*, 343–373. [CrossRef]
66. Halilu, A.S.; Waziri, M.Y. A transformed double step length method for solving large–scale systems of nonlinear equations. *J. Numeri. Math. Stoch.* **2017**, *9*, 20–32.
67. Waziri, M.Y.; Muhammad, L.; Sabi'u, J. A simple three-term conjugate gradient algorithm for solving symmetric systems of nonlinear equations. *Int. J. Adv. in Appl. Sci.* **2016**, *5*, 118–127. [CrossRef]
68. Birgin, E.G.; Martínez, J.M. A spectral conjugate gradient method for unconstrained optimization. *Appl Math. Optm* **2001**, *43*, 117–128. [CrossRef]
69. Dolan, E.D.; Moré, J.J. Benchmarking optimization software with performance profiles. *Math. Prog.* **2002**, *91*, 201–213. [CrossRef]
70. Yasrib, A.; Suhaimi, M.A. Image processing in medical applications, *J. Info. Tech.* **2003**, *3*, 63–68.
71. Collins, K.A.; Kielkopf, J.F.; Stassun, K.G.; Hessman, F.V. Image processing and photometric extraction for ultra-precise astronomical light curves. *The Astro. J.* **2017**, *153*, 177. [CrossRef]
72. Mishra, R.; Mittal, N.; Khatri, S.K. Digital image restoration using image filtering techniques. *IEEE Int. Conf. Autom. Comput. Tech. Manag.* **2019**, *6*, 268–272.
73. Sun, S.; He, T.; Chen, Z. Semantic structured image coding framework for multiple intelligent applications. *IEEE Trans. Cir. Syst. Video Tech.* **2021**, *31*, 3631–3642. [CrossRef]
74. Bruckstein, A.M.; Donoho, D.L.; Elad, M. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Rev.* **2009**, *51*, 34–81. [CrossRef]
75. Figueiredo, M.A.; Nowak, R.D. An EM algorithm for wavelet–based image restoration. *IEEE Trans. Image Process* **2003**, *12*, 906–916. [CrossRef]
76. Hale, E.T.; Yin, W.; Zhang, Y. *A Fixed–Point Continuation Method for (l_1)–Regularized Minimization with Applications to Compressed Sensing*; Technical Report TR07–07; Rice University: Houston, TX, USA, 2007; Volume 43, 44p.

77. Figueiredo, M.A.; Nowak, R.D.; Wright, S.J. Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems, *IEEE J. Sel. Top. Signal Process* **2007**, *1*, 586–597. [CrossRef]
78. Van Den Berg, E.; Friedlander, M.P. Probing the Pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.* **2008**, *31*, 890–912. [CrossRef]
79. Beck, A.; Teboulle, M. A fast iterative shrinkage–thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.* **2009**, *2*, 183–202. [CrossRef]
80. Hager, W.W.; Phan, D.T.; Zhang, H. Gradient–based methods for sparse recovery, *SIAM J. Imaging Sci.* **2011**, *4*, 146–165.
81. Awwal, A.M.; Kumam, P.; Mohammad, H.; Watthayu, W.; Abubakar, A.B. A Perry–type derivative–free algorithm for solving nonlinear system of equations and minimizing l_1 regularized problem. *Optimization* **2021**, *70*, 1231–1259. [CrossRef]
82. Ibrahim, A.H.; Deepho, J.; Abubakar, A.B.; Adamu, A. A three–term Polak–Ribière–Polyak derivative–free method and its application to image restoration. *Sci. Afri.* **2021**, *13*, e00880. [CrossRef]